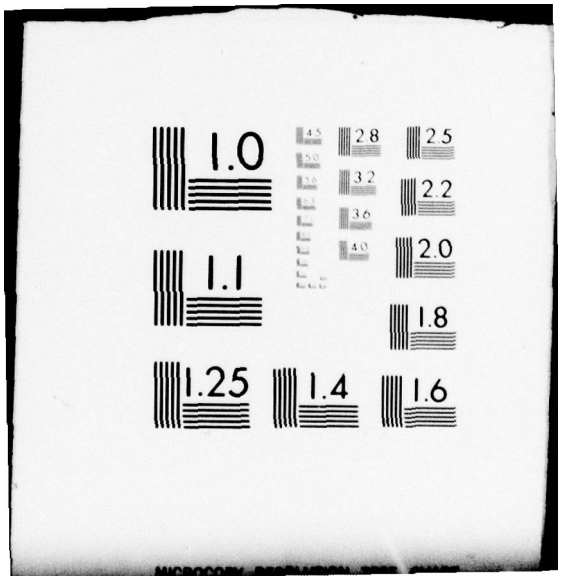
AD-A074 427 GTE SYLVANIA INC NEEDHAM HEIGHTS MASS ELECTRONIC SYS--ETC F/G 17/2
SUBBAND CODER STUDY.(U)
JUL 79 R S CHEUNG, R L WINSLOW DCA100-79-C-0001
UNCLASSIFIED SBIE-AD-E100 278 NL

1 OF 2
AD A074427


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION

FINAL REPORT

SUBBAND CODER STUDY
DCA 100-79-C-0001

THIS REPORT HAS BEEN PREPARED BY:

RONALD S. CHEUNG

RAIMOND L. WINSLOW

JULY 2, 1979

GTE SYLVANIA
INCORPORATED
ELECTRONIC SYSTEMS GROUP
EASTERN DIVISION

77 "A" STREET
NEEDHAM HEIGHTS, MASSACHUSETTS 02194

FINAL REPORT

SUBBAND CODER STUDY

DCA 100-79-C-0001

This report has been prepared by:

Ronald S. Cheung

Raimond L. Winslow

July 2, 1979

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
GTE Sylvania
77 "A" Street
Needham, MA 02194

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
(6) Subband Coder Study, (12) I28

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
(9) Final Report,

5. AUTHOR(S) (First name, middle initial, last name)
(10) Ronald S. Cheung
Raimond L. Winslow

406 451

6. REPORT DATE
(11) 2 July 79

7a. TOTAL NO. OF PAGES
125

7b. NO. OF REFS
16

8a. CONTRACT OR GRANT NO.
(15) DCA100-79-C-0001

8b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
(19) AD-E100 278
none

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited

DISTRIBUTION STATEMENT A
Approved for public release; Distribution Unlimited

11. SUPPLEMENTARY NOTES
(18) SBIE

12. SPONSORING MILITARY ACTIVITY
Defense Communications Agency
1860 Wiehle Avenue
Reston, VA 11090

13. ABSTRACT

This report gives a detailed treatment of the theory and implementation of a speech waveform coding technique known as subband coders whose operations include the partitioning of the input speech band into subbands and the quantization of subbands individually. Both the integer-band sampling and the quadrature mirror filtering (QMF) approaches are studied and the result indicates that the latter method yields much improved speech quality as compared to the former one.

In addition to the design of bandpass filters and subband quantizers, practical aspects in the implementation of subband coders are considered and they include

(1) a more efficient digital interpolation scheme, and
(2) a high speed multiplier-accumulator.

DD FORM 1 NOV 68 1473

12

Security Classification

AGO 5698A

406 451

14 KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
Subband Coding						
Integer-band Sampling Technique						
Digital Decimation						
Digital Interpolation						
Quadrature Mirror Filters						
Adaptive Bit Allocations						
Robust Quantizers						
Multiplier-Accumulator						

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I
Summary

1.1 Summary of the Program

Under the six-month Subband Coder Study Contract, (DCA 100-79-C-0001), GTE Sylvania developed real-time software for a full-duplex 9.6 Kb/s and a half-duplex 16.0 Kb/s speech transmission scheme using the subband encoding technique. This software was designed to operate on the two GTE Sylvania Programmable Signal Processors (PSP) already owned by the Defense Communications Agency.

This study and implementation effort has resulted in a number of significant accomplishments in developing speech processing algorithms and hardware. Among them, the most important ones are:

1) the development of real-time software for the 9.6 and 16.0 Kb/s subband coders
2) the development of a high speed programmable multiplier-accumulator (MULACC)
3) the development of a novel digital interpolation scheme
4) the development of an improved subband coder that employs quadrature mirror filters (QMF)

The Subband Coder Study was motivated by initial Bell Laboratories simulations which showed that the algorithm has potential of providing good performance with reasonable complexity at data rates between 9.6 and 16.0 Kb/s. Preliminary listening tests indicated that the subband coder yielded better speech quality than the continuously variable slope delta modulation (CVSD) of the same data rate. However, to evaluate the algorithm fully, a large number of processed sentences is necessary. Since it is impractical to generate such a data base using FORTRAN floating point simulations, real-time implementation of the subband coder algorithm on the PSP's provides a viable solution.

Though the subband coder is conceptually simple, programming of the algorithm on the PSP using fixed-point arithemtic is non-trivial based on the limitations of the machine's computational

speed and 16-bit accuracy. One such problem arises when the exact Bell Lab's band-pass filters have to be utilized in order to duplicate their simulations. These filters, in general, are characterized by their sharp cutoffs and flat passband responses. Unfortunately, each of them is 200 taps long and this increases tremendously the processing requirements. Furthermore, the fixed-point 16-bit arithmetic hardware in the PSP does not provide enough accuracy in performing the bandpass filtering operation due to intermediate product truncations and roundoffs. To circumvent the mentioned difficulties, GTE Sylvania designed a high-speed multiplier-accumulator (MULACC) which performs a 16 x 16 bit multiplication with a 35-bit product accumulation in 208 nsec (2 PSP machine cycles). Occupying two PC cards, this circuitry communicates to the PSP CPU via the I/O buses. One apparent advantage of the hardware is that the PSP's basic design remains unchanged and existing software is still operable on the modified machines. Moreover, the speed and accuracy of MULACC permits the real-time implementations of 9.6 and 16.0 Kb/s subband coders on the PSP's.

To further simplify the complexity of the algorithm, GTE Sylvania developed a new digital interpolation scheme which is capable of reducing the number of multiplications needed by 50%. Even though the method is not utilized in the real-time implementations, it can be directly applied to other signal processing areas that involve digital interpolations.

Upon the completion of the real-time programming, extensive listening tests were performed on the processed outputs. The results indicate that the speech quality of the subband coder is slightly "hollow" and noisy especially for the 9.6 Kb/s system, but its overall quality is significantly better than that of CVSD. However, when compared to other high quality 16 Kb/s speech encoding schemes, such as the Adaptive Predictive Coder with Adaptive Quantization (APCQ) developed for DCA under Contract DCA-100-76-C-0002, the subband coder is clearly inferior. This can be partly attributed to the fact that the four or five band-pass filters employed by the subband coder algorithm are not

adequate to represent the entire speech spectrum. This introduces spectral "gaps" which manifest the "hollowness" in the processed speech. Moreover, the ratio of subband energies fluctuates largely from frame to frame and quantization of them using adaptive PCM (APCM) with fixed bit allocations is suboptimal.

Armed with this new information, GTE Sylvania developed a FORTRAN simulation of an improved subband coding scheme using quadrature mirror filters (QMF). Utilizing a 3-stage QMF structure, the input speech band is split into 8 subbands which can be recombined perfectly to form the original one. No spectral distortions are introduced in the bandsplitting and reconstruction processes even with a relatively short filter. Furthermore, to enable a more efficient encoding of the subband waveforms, an adaptive bit allocation scheme is incorporated which assigns different quantizer bits according to the energies of the subbands. As compared to the original subband coder, FORTRAN simulations of the QMF approach indicate that the processed speech is much improved and especially at 16 Kb/s, high quality speech is resulted.

Because of the improved voice quality provided by the QMF subband coder, further work should be performed to refine the algorithm and implement it in real-time on the modified PSP's.

SECTION II

SOFTWARE SIMULATIONS

2.1 Introduction

The subband coding algorithms investigated under this contract are generally classified as waveform coding techniques which, in contrast to source encoding methods that model only the sources of speech production, utilize speech waveform properties, such as, distributions of amplitude and power, nonflat characteristics of speech spectra. Moreover, though quantization is applied to actual time waveforms, subband coding is categorized as a frequency domain technique [1]. This is due to the fact that the input speech band is partitioned into several frequency components known as subbands and they are individually encoded. By varying dynamically the number of bits utilized to quantize each subband, the overall available bits can be distributed such that the important frequency component is quantized with more accuracy. In the case of speech signals, perceptually important acoustic attributes, such as pitch and first formant which are located at low frequency subbands, are preferentially encoded with more bits. However, for upper subbands where fricatives or unvoiced sounds occur, encoders with fewer bits/sample are utilized since these subbands contribute less perceptually. An obvious advantage of subband coding over coding in full band is that different quantizers are employed. As a result, the quantization noise is confined to each individual subband and this prevents the masking of signals in one frequency band by quantization noise of another.

2.2 Operating Principles of Subband Coder

Operations of the subband coder include the partitioning of the input speech band into N subbands using bandpass filters, H_1, H_2, ... H_N in the manner shown in Figure 2-1. Each of the resulting

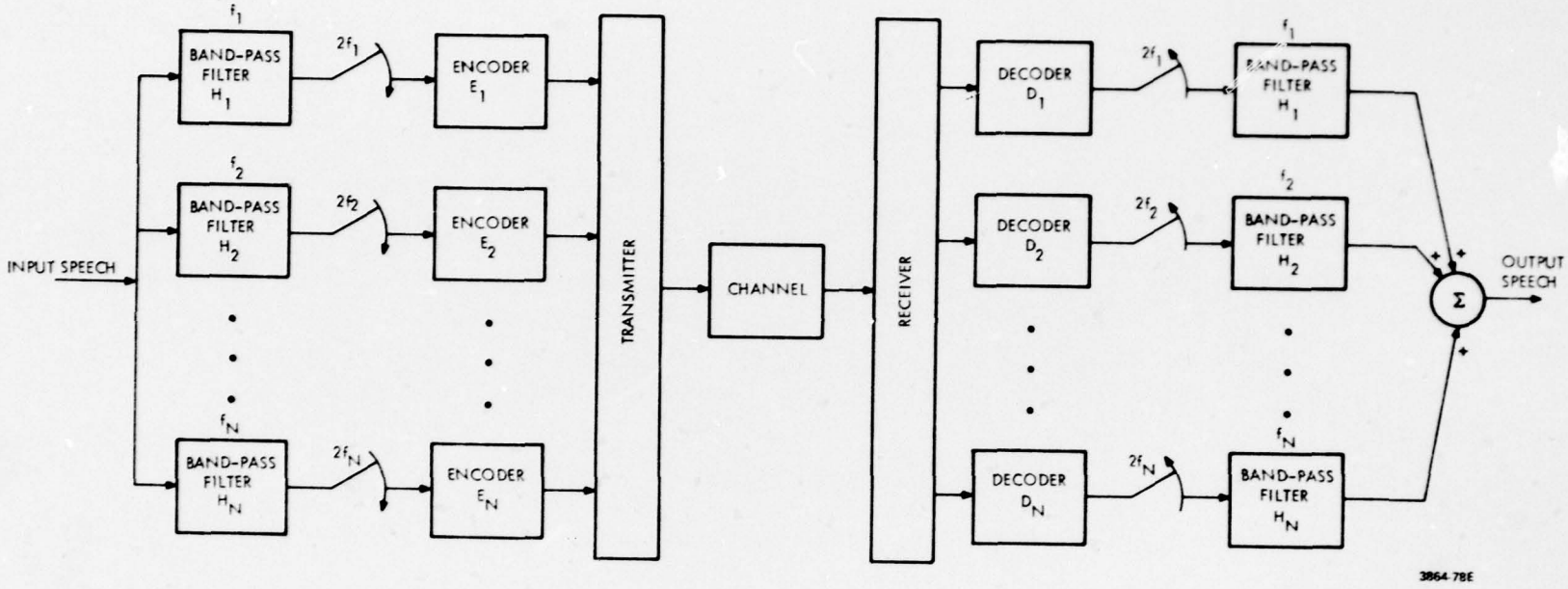


FIGURE 2-1 BLOCK DIAGRAM OF THE SUBBAND CODER

subbands, after frequency translation to DC, is digitally encoded for transmission. At the receiver, these transmitted data are decoded and later modulated upwards to their corresponding center frequencies. After bandpass filtering, the resulting subbands are then added together to form an estimate of the input signal [2].

2.2.1 Integer-Band Sampling

To perform the frequency translations of the subband spectra, generally a complicated modulation/demodulation scheme has to be applied. Fortunately, through a new technique known as integer-band sampling, this modulation process can be avoided [3]. This method calls for the filtering of the input signal into subbands of bandwidths W_i followed by the resampling of the outputs at their corresponding Nyquist rates $2W_i$. At the receiver, the reverse operation is performed which reconstructs the input waveform through upsampling the decoded subbands followed by bandpass filtering. Though the integerband sampling technique alleviates the use of any complex modulation schemes, the choice of subbands is rather restricted. Moreover, to prevent spectral aliasing, the subbands have to lie between frequencies kW_i and $(k+1)W_i$ where k is an integer.

For explanatory purposes, the bandpass filtering of one subband is detailed in Figure 2-2(a). If y(n) denotes the bandpass signal sampled originally at $f_s = \frac{1}{T}$, and $Y(e^{jWT})$, the spectrum of y(n), is depicted in Figure 2-2(b). The resampled signal $y_1(n)$ is given by:

$$y_1(n) = y(Mn) = y\left(\frac{f_s}{2W_i} n\right) \qquad (2\text{-}1)$$

where $M = \frac{f_s}{2W_i}$ is defined as the decimation ratio. Consequently, $y_1(n)$ has a lower sampling rate $f'_s = \frac{1}{T'} = \frac{f_s}{M}$.

Defining a new sequence w(n) as:

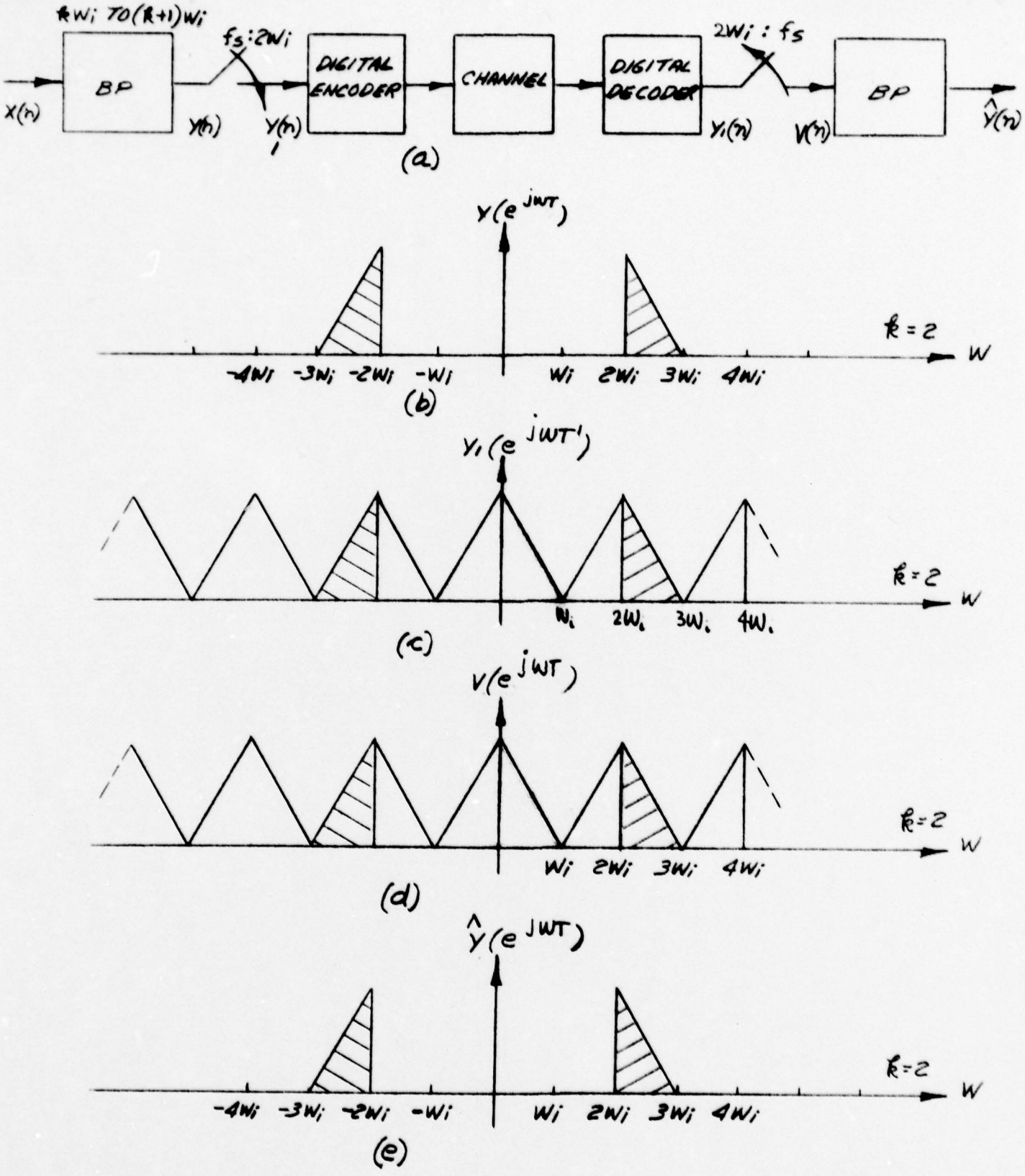


FIGURE 2-2 INTEGER-BAND SAMPLING AND A FREQUENCY DOMAIN INTERPRETATION

$$w(n)=\begin{cases} y(n) & ; \quad n=0, \pm M, \pm 2M, \ldots \\ 0 & ; \quad \text{otherwise} \end{cases} \tag{2-2}$$

$$\text{or} \qquad w(n) = \frac{y(n)}{M} \sum_{\ell=0}^{M-1} e^{j \frac{2\pi \ell n}{M}} \tag{2-3}$$

The sequence $y_1(n)$ can be rewritten as:

$$y_1(n) = w(Mn) \tag{2-4}$$

To compare the spectra of $y_1(n)$ and $y(n)$, the Z-transform of Eq. (2-4) is taken which yields [4]:

$$\begin{aligned} Y_1(Z) &= \sum_{n=-\infty}^{\infty} w(Mn) Z^{-n} \\ &= \sum_{n=-\infty}^{\infty} w(n) Z^{n/M} \\ &= \frac{1}{M} \sum_{\ell=0}^{M-1} \left(\sum_{n=-\infty}^{\infty} y(n) e^{j \frac{2\pi \ell n}{M}} Z^{-n/M} \right) \\ &= \frac{1}{M} \sum_{\ell=0}^{M-1} Y\left(e^{-j \frac{2\pi \ell}{M}} Z^{1/M} \right) \end{aligned} \tag{2-5}$$

The Fourier transform is obtained by evaluating the Z-transform shown in Eq. (2-5) on the unit circle and this results in:

$$Y_1\left(e^{jwT'}\right) = \frac{1}{M}\sum_{\ell=0}^{M-1} Y\left(e^{j(wT'-2\pi\ell)/M}\right) \qquad (2\text{-}6)$$

From Eq. (2-6), $Y_1(e^{jwT'})$ depicts a repeated version of $Y(e^{jwT'})$ and a graphical representation of it is shown in Figure 2-2(c). To prevent spectral aliasing, it is clear that the subband frequency has to lie between kW_i and $(k+1)W_i$ where k is an integer (k=2 in Figure 2-2). This imposes a constraint on the choice of subbands which affects the performance of subband coders tremendously.

At the receiver, the decimated signal $y_1(n)$ is interpolated upwards M times in order to achieve the original sampling frequency of $f_s = \frac{1}{T}$. In fact, every (M-1) zeros are inserted between consecutive y(n)'s and the resulting sequence v(n) is defined as:

$$v(n) = \begin{cases} y_1(n/M) & ; \quad n=0, \pm M, \pm 2M, \ldots \\ \\ \phi & ; \quad \text{otherwise} \end{cases} \qquad (2\text{-}7)$$

To see the relationships between the spectra of v(n) and $y_1(n)$, take the Z-transform of v(n) and this yields:

$$V(Z) = \sum_{n=-\infty}^{\infty} y_1(n/M)Z^{-n} = \sum_{n=-\infty}^{\infty} y_1(n)Z^{-Mn} \qquad (2\text{-}8)$$

$$= Y_1(Z^m)$$

The Fourier transform of Eq. (2-8) is:

$$V\left(e^{jwT}\right) = Y_1\left(e^{jwTM}\right) = Y_1\left(e^{jwT'}\right) \qquad (2\text{-}9)$$

where T' is the sampling period of the decimated waveform $y_1(n)$. A graphical representation of $V(e^{jwT})$ is shown in Figure 2-2(d).

From Eq. (2-9), the spectrum of v(n) has the identical shape as that of $y_1(n)$ after normalization with respect to their sampling rates. By bandpass filtering v(n), a replica of y (n) is created as shown in Figure 2-2(e).

2.2.2 Characteristics of Band Pass Filters

Utilizing the integer-band sampling technique, the dividing of input speech band into subbands is accomplished with a set of bandpass filters whose passbands are chosen to span the entire speech band with minimal overlap. Specifications of 9.6 and 16.0 Kb/s subband filters and quantizer parameters are summarized in Tables2-1 and 2-2. For the 9.6 Kb/s system, there are four subbands which cover ranges from 240-2880 Hz. Each of the filters is 200 taps long and is characterized by a relatively flat passband and a short transition region. The coefficients of the bandpass filters, furnished by R. Crochiere of Bell Labs, are shown in Appendix A. The overall magnitude response of the 9.6 Kb/s system, obtained by combining the responses of the 4 filters, is shown in Figure 2-3 and it reveals large spectral gaps at frequencies around 1000 Hz and 1800 Hz which accounts for the "hollow" and reverberent quality in the processed speech. As for the 16.0 Kb/s system, five subbands which span frequencies from 178-3200 Hz are utilized. These filters are also 200 taps long and their coefficients are tabulated in Appendix B. The overall magnitude response, obtained by combining the responses of five filters, is shown in Figure 2-4. As compared to that of the 9.6 Kb/s system, Figure 2-4 shows a narrowing of the gaps in the reconstructed spectrum which results in improved speech quality.

Sampling Rate = 9.6 KHz

Data Rate = 9.6 Kb/s

No. of Input Samples/Frame = 180

Frame Time = 18.75 msec

SUBBAND NO.	BAND EDGES (HZ)	DECIMATION (INTERPOLATION) RATIO	NO. OF DECIMATED SAMPLES/FRAME	QUANTIZER BITS	NO. OF SUBBAND BITS/FRAME	MIN STEP SIZE ΔMIN	MAX STEP SIZE ΔMAX	STEP SIZE THRESHOLD ΔTH
1	240- 480	20	9	3	27	5	1280	5
2	480- 960	10	18	2	36	6	1536	6
3	1067-1600	9	20	2	40	6	1536	6
4	1920-2880	5	36	2	72	7	1792	7
Synchronization					5			
TOTAL					180			

TABLE 2-1 SPECIFICATIONS OF 9.6 KB/S SUBBAND CODER

Sampling Rate = 10.6667 KHz

Data Rate = 16.0 KB/s

No. of Input Samples/Frame = 90

Frame Time = 8.4375 msec

SUBBAND NO.	BAND EDGES (HZ)	DECIMATION (INTERPOLATION) RATIO	NO. OF DECIMATED SAMPLES/FRAME	QUANTIZER BITS	NO. OF SUBBAND BITS/FRAME	MIN STEP SIZE ΔMIN	MAX STEP SIZE ΔMAX	STEP SIZE THRESHOLD ΔTH
1	178- 356	30	3	4	12	5	1280	0
2	296- 593	18	5	4	20	5	1280	0
3	533-1067	10	9	3	27	6	1536	0
4	1067-2133	5	18	2	36	6	1536	7
5	2133-3200	5	18	2	36	7	1792	7
Synchronization					4			
TOTAL					135			

TABLE 2-2 SPECIFICATIONS OF 16.0 KB/S SUBBAND CODER

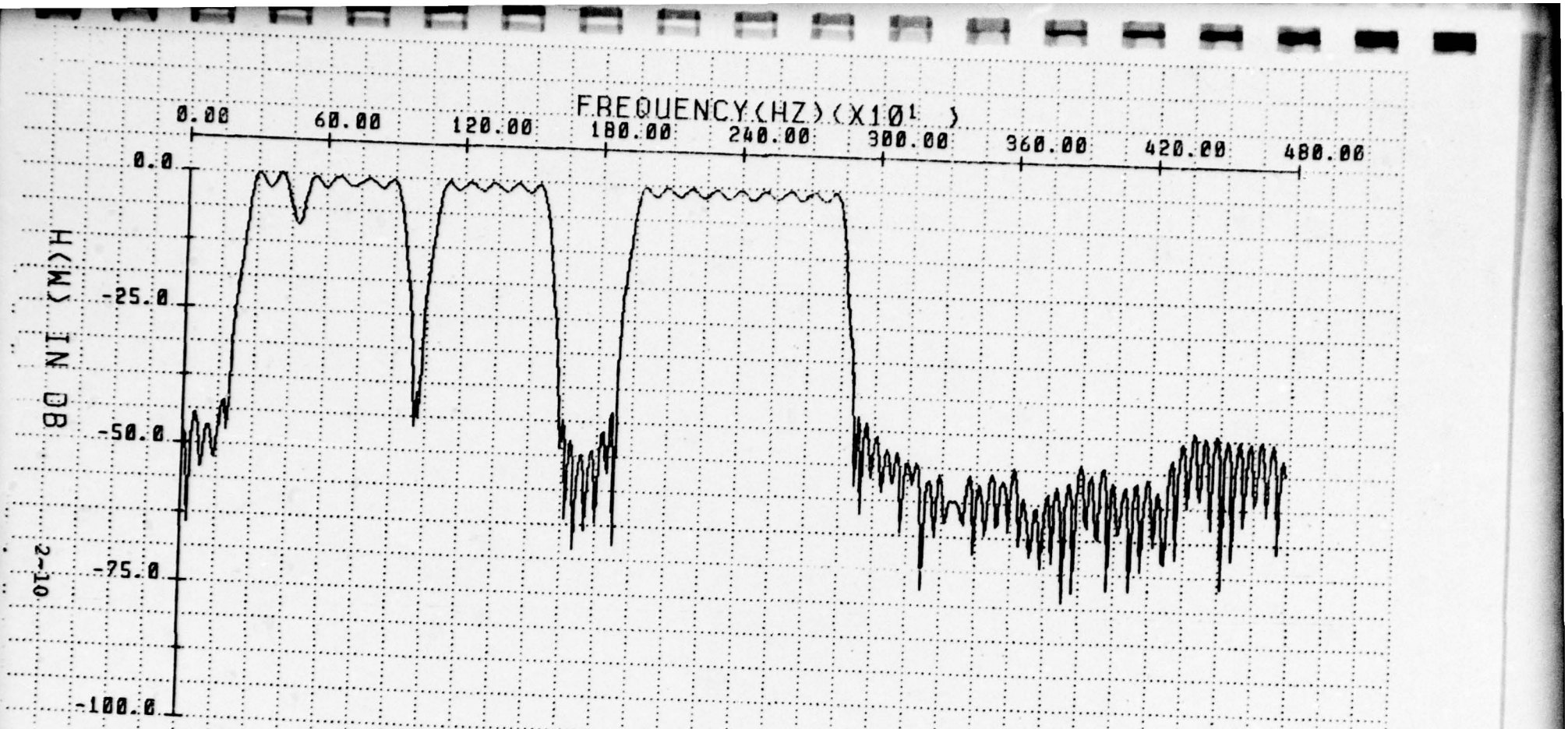


FIGURE 2-3 SPECTRAL RESPONSE OF 9.6 KBPS SYSTEM

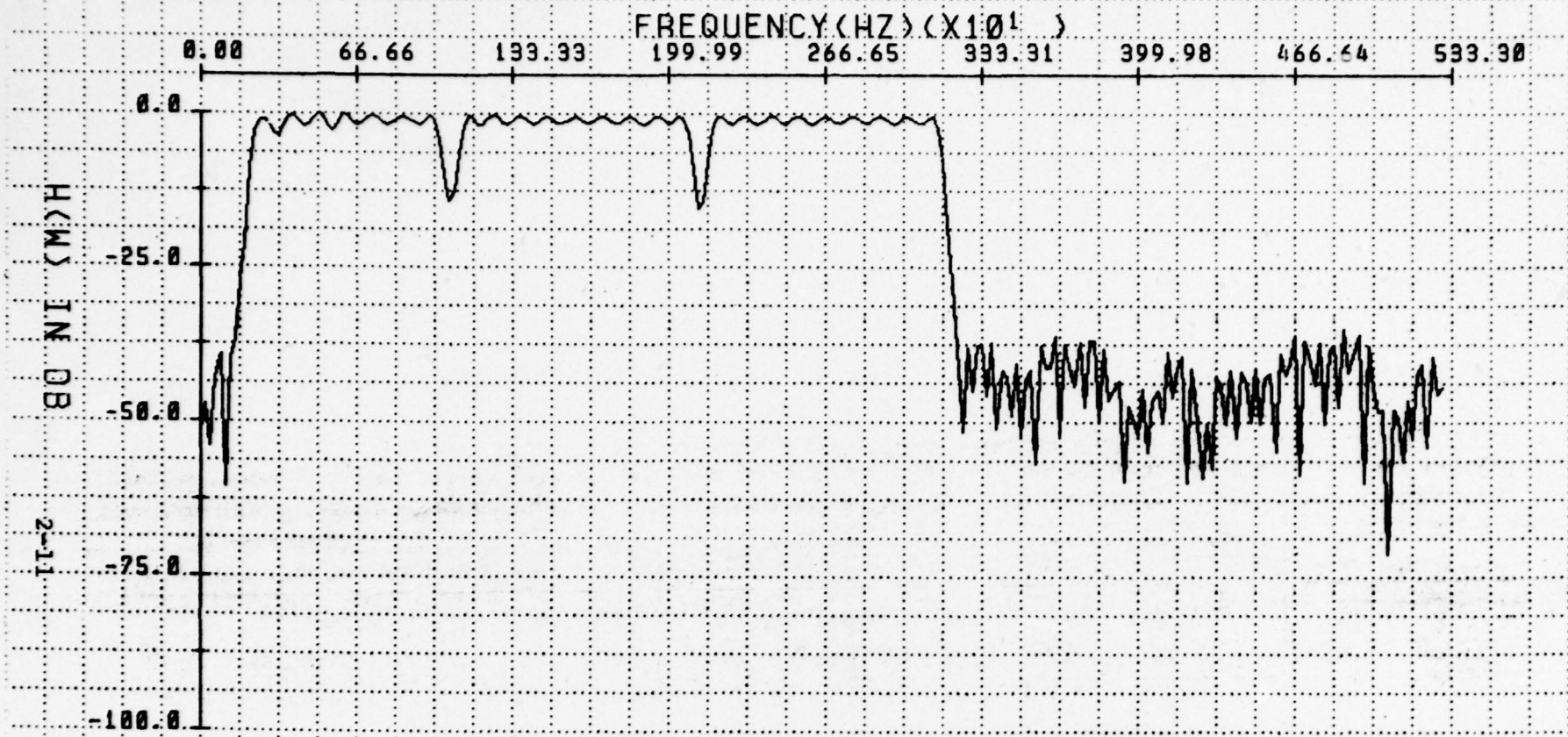


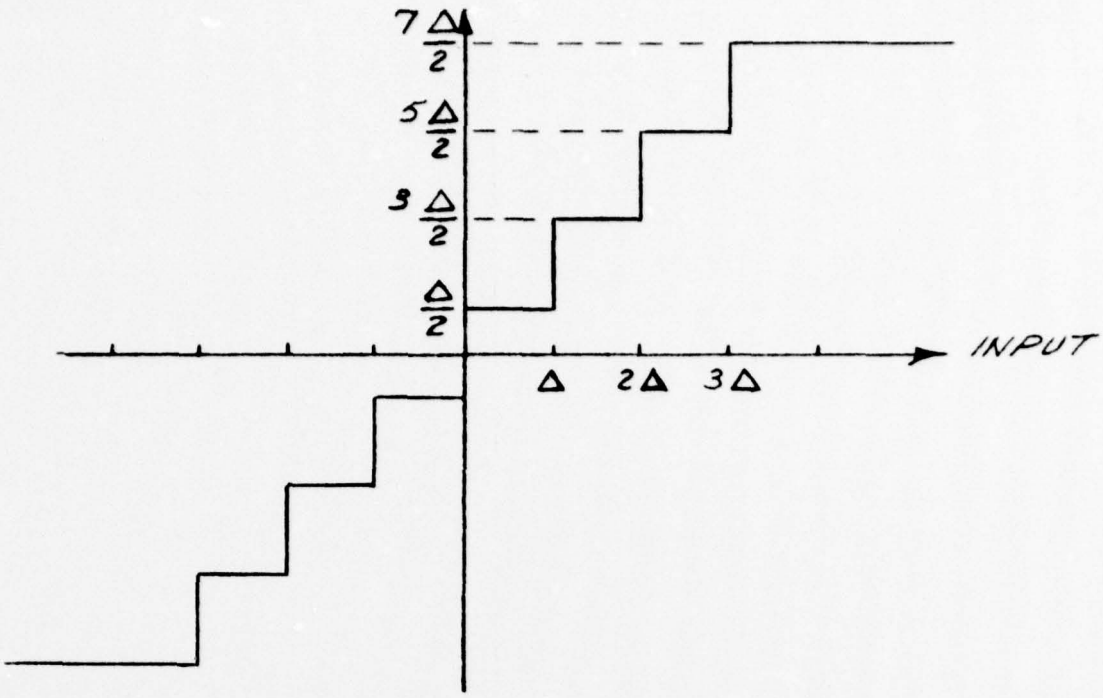
FIGURE 2-4 SYSTEM RESPONSE OF 16-KBPS SUB-BAND CODER

2.2.3 Encoding of Subband Signals

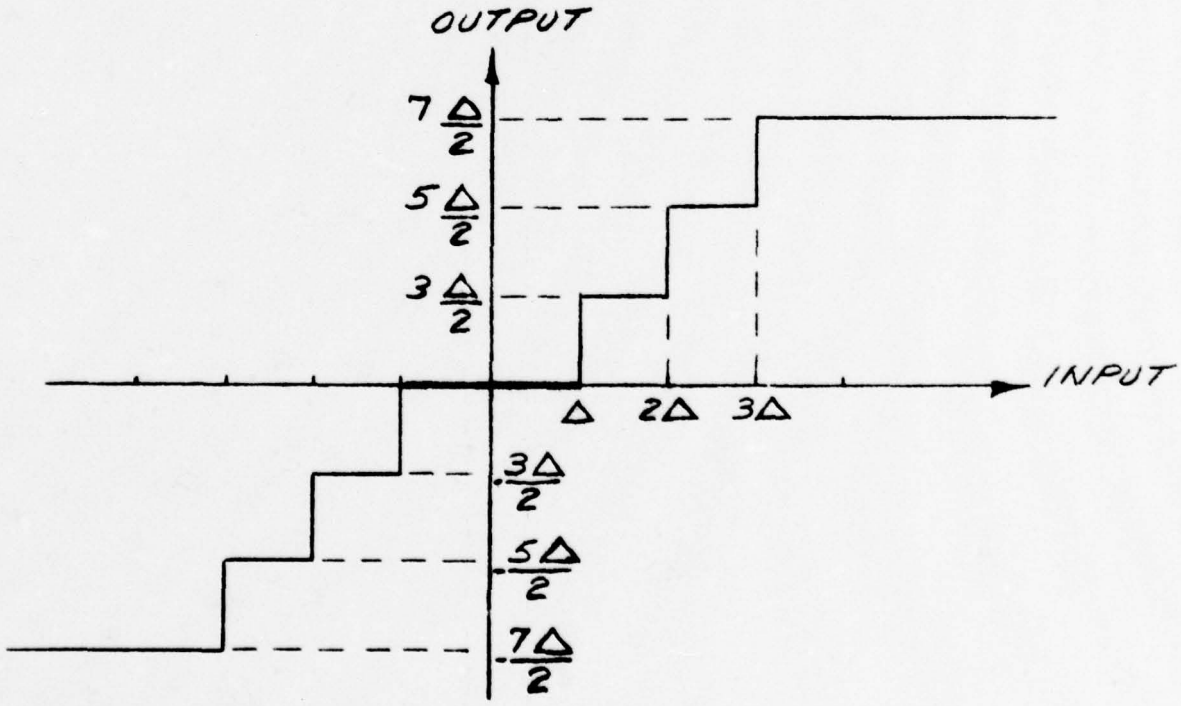
It is well known that the long-term spectrum of speech signals tends to decrease in power density with frequency, yet the drop is relatively small for waveforms across any one subband. In other words, after frequency translation of the bandpassed input to DC and resampling it at the Nyquist rate, the signal has little correlation between adjacent samples and encoding of it was shown to be best performed using adaptive PCM (APCM)[3]. The most common scheme is the step-size adaptation strategy which utilizes one-word step size memory[5]. This technique is dealt with in great depth in recent literature and it will not be treated in detail here. Basically, a B-bit quantizer (where B is an integer) is employed to encode the input sample into one of the possible 2^B LEVELS. Then the quantizer updates its step size after every input sample. In general, these quantizers are referred to as mid-rise quantizers and an example is a 3-bit quantizer whose characteristic is shown in Figure 2-5(a). Also, multiplication factors for 2, 3, and 4-bit quantizers are listed in Table 2-3.

In practice, minimum (Δmin) and maximum (Δmax) values have to be imposed on the step sizes to assure proper quantizer adaptation. Hence, an obvious disadvantage of mid-rise quantizers is that it cannot represent a zero level. For small amplitude inputs, the quantized values always oscillate between $\pm\Delta$min which are perceived as a low level tone. In the case of subband coders, this low level tone will be frequency translated back in the middle of the speech band where they are particularly noticeable. Hence, even with a small Δmin, the distortion is still very disturbing.

One method to alleviate the low volume tones is to make use of the mid-tread quantizer[7]. An example is the 7-level quantizer whose characteristic is shown in Figure 2-5(b). Since the quantizer has a zeroth step, all low level



(a) THE 8-LEVEL MID-RISE QUANTIZER CHARACTERISTIC



(b) THE 7-LEVEL MID-TREAD QUANTIZER CHARACTERISTIC

FIGURE 2-5 QUANTIZER CHARACTERISTICS

inputs are then coded to be zero and this eliminates the tones. Hence, mid-tread quantizers seem advantageous for subband coders. However, in contrast to a mid-rise one, a B-bit mid-tread quantizer has less than 2^B levels and this represents a decrease in coding efficiency. For subband coders, mid-tread quantizers can still be employed without losing much efficiency. This is accomplished by utilizing a mid-rise/mid-tread dequantizer switch at the receiver. In this case, the transmitter employs conventional B-bit mid-rise quantizers whereas the receiver can select either mid-tread or mid-rise to dequantize the data. As a matter of fact, a step-size threshold value (Δth) is setup in such a way that a mid-tread dequantizer is selected if $\Delta < \Delta th$ whereas the mid-rise dequantizer is utilized for $\Delta > \Delta th$. Threshold values for the 9.6 and 16.0 Kb/s subband coders are experimentally determined and they are included in Tables 2-1 and 2-2.

STEP SIZE MULTIPLICATION FACTOR	QUANTIZER BITS		
	4	3	2
M_1	0.9	0.85	0.85
M_2	0.9	1.0	1.9
M_3	0.9	1.0	
M_4	0.9	1.5	
M_5	1.2		
M_6	1.6		
M_7	2.0		
M_8	2.4		

TABLE 2-3 APCM CODER PARAMETERS

2.3 Practical Considerations in the Implementation of Subband Coders

The subband coders as discussed in Section 2.2 employ a bank of 4 or 5 FIR filters to perform bandpass filtering. One

obvious advantage is due to the linearity of the FIR filter's phase response with respect to frequency which eliminates distortions introduced by time delays of different subbands. However, to achieve the desired bandpass characteristics, length of FIR filters is generally long (e.g., 200 taps) thus increasing the processing requirements. In the real-time implementations of subband coders, the success is greatly dependent on the efficient programming of the bandpass filtering operation and the following sections consider in more detail the number of mathematical operations (in particular, the number of multiplies) required.

2.3.1 Decimation Filtering

The outputs of a bandpass filter is given by the convolution-sum

$$y(n)=\sum_{k=0}^{N-1} h(k)x(n-k) \qquad (2\text{-}10)$$

where N is the order of the filter, h(k) is the k^{th} filter coefficient, and x(n), y(n) represent the input, output sample, respectively. It is clear from Eq. (2-10) that N multiplications are needed to compute each y(n). However, the integer-band sampling technique calls for the down-sampling of the filtered output to obtain $y_1(n)$. In other words, only every M^{th} point has to be calculated where M is the decimation ratio and Eq. (2-10) can be re-written as:

$$y_1(n) = y(Mn) = \sum_{k=o}^{N-1} h(k)x(Mn-k) \qquad (2\text{-}11)$$

Henceforth, the total number of multiplies required for each input sample becomes N/M. Furthermore, since the impulse response of a FIR filter satisfies:

$$h(n) = h(N-1-n) \tag{2-12}$$

Eq. (2-11) can be shown as (for N even):

$$y_1(n) = y(Mn) = \sum_{k=o}^{\frac{N}{2}-1} h(k) \Big(x(Mn-k) + x(Mn-N+k+1) \Big) \tag{2-13}$$

Then the symmetry of the filter reduces the number of multiplies by 50% (or N/2M per input sample). The above derivation can also be applied to the case when N is odd. Figure 2-6 illustrates the structure that exploits the symmetry property of the FIR filters. Table 2-4 summarizes the number of multiplies needed for each band to perform decimation filtering in the 9.6 and 16.0 Kb/s subband coders.

2.3.2 Interpolation Filtering

There are a number of methods which may be used to perform interpolation filtering. The most straightforward method is to pad in (M-1) zeros between each decimated sample value where M in this case the interpolation ratio, and bandpass filter the resulting sequence. Using this technique, the filter structure as shown in Figure 2-6 is also capable of performing interpolation filtering. Though the symmetry of linear phase FIR filters can be employed to decrease the number of multiplies, the amount of multiplications required (N/2 per input sample) is still excessive. However, cognizant of the fact that there are no multiplications involved with samples of zero values, this number can be greatly reduced. Referring to Figure 2-2(a), the synthesized sample $\hat{y}(n)$ is shown as

$$\hat{y}(n) = \sum_{k=o}^{N-1} h(k)y(n-k) \tag{2-14}$$

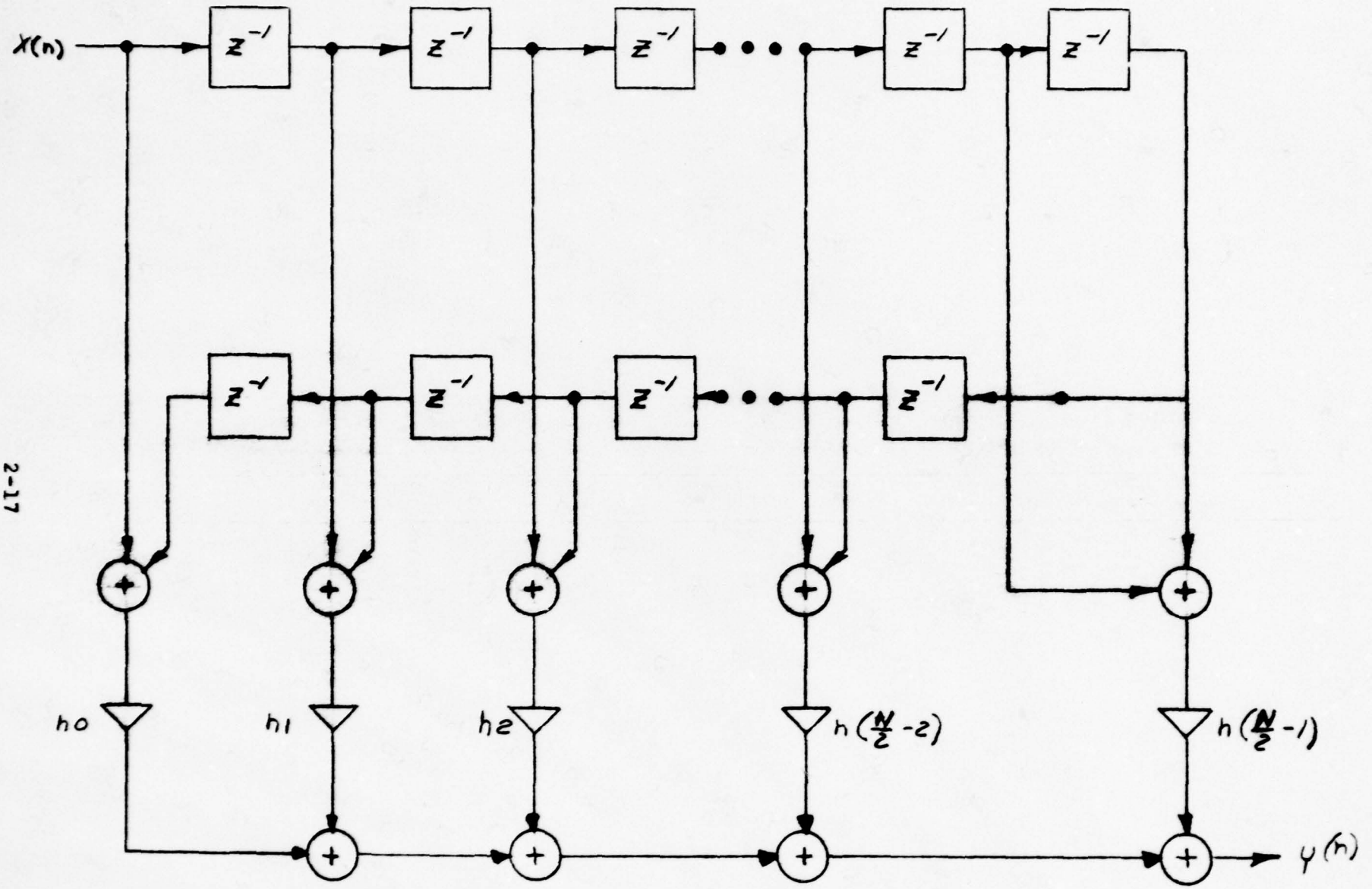


FIG 2-6 DECIMATION FILTER STRUCTURE (N EVEN)

SUBBAND #	# OF MULTIPLIES PER FRAME IN 9.6 KB/S SYSTEM	# OF MULTIPLIES PER FRAME IN 16.0 KB/S SYSTEM
1	900	300
2	1800	500
3	2000	900
4	3600	1800
5		1800
TOTAL	8300	5300

TABLE 2-4 NUMBER OF MULTIPLIES PER FRAME FOR DIGITAL DECIMATION

Substituting Eq. (2-7) into (2-14), $\hat{y}(n)$ can be rewritten as

$$\hat{y}(n) = \sum_{k=o}^{N-1} h(k) y_1 \left[\frac{n-k}{M} \right] \tag{2-15}$$

where

$$\left[\xi\right] = \begin{cases} \xi; \text{ if } \xi \text{ is an integer} \\ 0; \text{ otherwise} \end{cases} \tag{2-16}$$

Using this technique, the number of multiplies required per input sample as shown in Eq. (2-15) is **equal to the integer quotient** {N/M}. The total number of multiplies per frame for interpolation filtering in the subband coders is shown in Table 2-5.

2.3.3 A Novel Technique in Digital Interpolation

If Table 2-5 is compared to Table 2-4, it is shown that the interpolation filtering requires twice the number of multiplies as that of decimation filtering. This can be attributed to the fact that the symmetry of FIR filters cannot be exploited in the interpolation case [4]. However, by rearranging and combining some of the input samples, the symmetry of the filter can indeed be employed to reduce the total number of multiplies at the expense of more additions (subtractions).

Defining

$$m = \left\{ \frac{N-1}{M} \right\} \tag{2-17}$$

where $\{\cdot\}$ is the integer quotient, N is the filter length and M is the decimation ratio, the remainder α can be written as:

$$\alpha = N-1 - mM \tag{2-18}$$

SUBBAND #	# OF MULTIPLIES PER FRAME REQUIRED IN 9.6 KB/S SYSTEM	# OF MULTIPLIES PER FRAME REQUIRED IN 16.0 KB/S SYSTEM
1	1,800	630
2	3,600	1,080
3	4,000	1,800
4	7,200	3,600
5		3,600
TOTAL	16,600	10,710

TABLE 2-5 NUMBER OF MULTIPLIES PER FRAME FOR INTERPOLATION FILTERING

Then Eq. (2-15), for the first M samples, can be expanded as:

$$\hat{y}(0) = h(0)y_1(0) + h(M)y_1(-1) + \cdots + h(mM-M)y_1(-m+1) + h(mM)y_1(-m)$$

$$\hat{y}(1) = h(1)y_1(0) + h(M+1)y_1(-1) + \cdots + h(mM-M+1)y_1(-m+1) + h(mM+1)y_1(-m)$$

$$\vdots$$

$$\hat{y}(\alpha) = h(\alpha)y_1(0) + h(M+\alpha)y_1(-1) + \cdots + h(mM-M+\alpha)y_1(-m+1) + h(mM+\alpha)y_1(-m)$$

$$\hat{y}(\alpha+1) = h(\alpha+1)y_1(0) + h(M+\alpha+1)y_1(-1) + \cdots + h(mM-M+\alpha+1)y_1(-m+1)$$

$$\hat{y}(\alpha+2) = h(\alpha+2)y_1(0) + h(M+\alpha+2)y_1(-1) + \cdots + h(mM-M+\alpha+2)y_1(-m+1) \qquad (2\text{-}19)$$

$$\vdots$$

$$\hat{y}(M-1) = h(M-1)y_1(0) + h(2M-1)y_1(-1) + \cdots + h(mM-1)y_1(-m+1)$$

Making use of the symmetry of h(n) as defined in Eq. (2-12),

$$\begin{aligned} h(mM) &= h(\alpha) \\ h(mM-M) &= h(\alpha+M) \\ &\vdots \\ h(0) &= h(N-1) \end{aligned} \qquad (2\text{-}20)$$

Substituting Eq. (2-20) into (2-19), the following equations are resulted:

$$\hat{y}(0) = h(0)y_1(0) + h(M)y_1(-1) + \cdots + h(\alpha+M)y_1(-m+1) + h(\alpha)y_1(-m)$$

$$\hat{y}(1) = h(1)y_1(0) + h(M+1)y_1(-1) + \cdots + h(\alpha+M-1)y_1(-m+1) + h(\alpha-1)y_1(-m)$$

$$\vdots$$

$$\hat{y}(\alpha) = h(\alpha)y_1(0) + h(M+\alpha)y_1(-1) + \cdots + h(M)y_1(-m+1) + h(0)y_1(-m)$$

$$\hat{y}(\alpha+1) = h(\alpha+1)y_1(0) + h(M+\alpha+1)y_1(-1) + \cdots + h(M-1)y_1(-m+1) \qquad (2\text{-}21)$$

$$\hat{y}(\alpha+2) = h(\alpha+2)y_1(0) + h(M+\alpha+2)y_1(-1) + \cdots + h(M-2)y_1(-m+1)$$

$$\vdots$$

$$y(M-1) = h(M-1)y_1(0) + h(2M-1)y_1(-1) + \cdots + h(\alpha+1)y_1(-m+1)$$

A closer examination of Eq. (2-21) reveals great similarity between sample pairs $\hat{y}(0)$ and $\hat{y}(\alpha)$; $\hat{y}(1)$ and $\hat{y}(\alpha-1)$ ... In other words, if they are combined, the following is true:

$$\begin{cases} a_0 \triangleq \frac{1}{2}(\hat{y}(0)+\hat{y}(\alpha)) = \frac{(h(0)+h(\alpha))}{2}(y_1(0)+y_1(-m)) + (\frac{h(M)+h(M+\alpha)}{2})(y_1(-1)+y_1(-m+1)) + \cdots \\ b_0 \triangleq \frac{1}{2}(\hat{y}(0)-\hat{y}(\alpha)) = \frac{(h(0)-h(\alpha))}{2}(y_1(0)-y_1(-m)) + (\frac{h(M)-h(M+\alpha)}{2})(y_1(-1)-y_1(-m+1)) + \cdots \end{cases}$$

$$\begin{cases} a_1 \triangleq \frac{1}{2}(\hat{y}(1)+\hat{y}(\alpha-1)) = (\frac{h(1)+h(\alpha-1)}{2})(y_1(0)+y_1(-m)) + (\frac{h(M+1)+h(M+\alpha-1)}{2})(y_1(-1)+y_1(-m+1)) + \cdots \\ b_1 \triangleq \frac{1}{2}(\hat{y}(1)-\hat{y}(\alpha-1)) = \frac{(h(1)-h(\alpha-1))}{2}(y_1(0)-y_1(-m)) + \frac{(h(M+1)-h(M+\alpha-1))}{2}(y_1(-1)-y_1(-m+1)) + \cdots \end{cases}$$

$$\vdots$$

$$a_{\{\frac{\alpha}{2}\}} \triangleq \frac{1}{2}\left(\hat{y}(\{\tfrac{\alpha}{2}\}) + \hat{y}(\alpha - \{\tfrac{\alpha}{2}\})\right) = \frac{\left(h(\{\frac{\alpha}{2}\}) + h(\alpha - \{\frac{\alpha}{2}\})\right)}{2}\left(y,10) + y,(-m)\right) + \cdots$$

$$b_{\{\frac{\alpha}{2}\}} \triangleq \frac{1}{2}\left(\hat{y}(\{\tfrac{\alpha}{2}\}) - \hat{y}(\alpha - \{\tfrac{\alpha}{2}\})\right) = \frac{\left(h(\{\frac{\alpha}{2}\}) - h(\alpha - \{\frac{\alpha}{2}\})\right)}{2}\left(y,10) - y,(-m)\right) + \cdots \quad (2\text{-}22)$$

where $\{\frac{\alpha}{2}\}$ denotes the largest integer of $\frac{\alpha}{2}$. Hence from Eq. (2-22),

$$\begin{aligned}
\hat{y}(0) &= a_0 + b_0 \\
\hat{y}(1) &= a_1 + b_1 \\
\hat{y}(2) &= a_2 + b_2 \\
&\vdots \\
y(\alpha-2) &= a_2 - b_2 \\
y(\alpha-1) &= a_1 - b_1 \\
y(\alpha) &= a_0 - b_0
\end{aligned} \quad (2\text{-}23)$$

Since it only takes $\left\{\frac{N-1}{2M}\right\}+1$ multiplies to obtain a_i or b_i, the total number of multiplies for each $\hat{y}(k)$ is also $\left\{\frac{N-1}{2M}\right\}+1$ which is about half the number required by the conventional method as shown in Eq. (2-15). Similarly, $\hat{y}(\alpha+1)$, $\hat{y}(M-1)$; $\hat{y}(\alpha+2)$, $\hat{y}(M-2)$; ... can be combined in the same fashion yielding:

$$a_{\{\frac{\alpha}{2}\}+1} \triangleq \tfrac{1}{2}\left(\hat{y}(\alpha+1)+\hat{y}(M-1)\right)=\frac{\left(h(\alpha+1)+h(M-1)\right)}{2}\left(y_1(0)+y_1(-m+1)\right)+\frac{\left(h(M+\alpha+1)+h(2M-1)\right)}{2}\left(y_1(-1)+y_1(-m+2)\right)+\cdots$$

$$b_{\{\frac{\alpha}{2}\}+1} \triangleq \tfrac{1}{2}\left(\hat{y}(\alpha+1)-\hat{y}(M-1)\right)=\frac{\left(h(\alpha+1)-h(M-1)\right)}{2}\left(y_1(0)-y_1(-m+1)\right)+\frac{\left(h(M+\alpha+1)-h(2M-1)\right)}{2}\left(y_1(-1)-y_1(-m+2)\right)+\cdots \quad (2\text{-}24)$$

and

$$\begin{aligned}\hat{y}(\alpha+1) &= a_{\{\frac{\alpha}{2}\}+1} + b_{\{\frac{\alpha}{2}\}+1} \\ &\vdots \\ \hat{y}(M-1) &= a_{\{\frac{\alpha}{2}\}+1} - b_{\{\frac{\alpha}{2}\}+1}\end{aligned} \quad (2\text{-}25)$$

As shown in Eq. (2-25), it takes $\left\{\frac{N-1}{2M}\right\}$ multiplies to compute each $\hat{y}(k)$ where $k = \alpha+1, \alpha+2, \ldots M-1$ which is also half the number required by the conventional way. Then the total number of multiplies needed to compute all M points is given by

$$\begin{aligned}N_M &= \left(\left\{\frac{N-1}{2M}\right\}+1\right)(\alpha+1) + \left\{\frac{N-1}{2M}\right\}(M-1-\alpha) \\ &= \left\{\frac{N-1}{2M}\right\}M + (\alpha+1)\end{aligned} \quad (2\text{-}26)$$

As an example, the filter structure needed to compute a_o, b_o (or equivalently $\hat{y}(0)$, $\hat{y}(\alpha)$) is depicted in Figure 2-7. To compute a_1, b_1 (or $\hat{y}(1)$, $\hat{y}(\alpha-1)$, the same filter structure with a different coefficient set can be utilized.

Furthermore, to compute a new block of output data, $\hat{y}(k)$'s can be expanded as:

$$\hat{y}(M) = h(0)y_1(1) + h(M)y_1(0) + h(2M)y_1(-1) + \cdots + h(\alpha+M)y_1(-m+2) + h(\alpha)y_1(-m+1)$$

$$\hat{y}(M+1) = h(1)y_1(1) + h(M+1)y_1(0) + h(2M+1)y_1(-1) + \cdots + h(\alpha+M+1)y_1(-m+2) + h(\alpha-1)y_1(-m+1)$$

$$\vdots$$

$$\hat{y}(M+\alpha) = h(\alpha)y_1(1) + h(M+\alpha)y_1(0) + h(2M+\alpha)y_1(-1) + \cdots + h(M)y_1(-m+2) + h(0)y_1(-m+1)$$

$$\hat{y}(M+\alpha+1) = h(\alpha+1)y_1(1) + h(M+\alpha+1)y_1(0) + \cdots + h(M-1)y_1(-m+2)$$

$$\vdots \qquad (2\text{-}27)$$

$$\hat{y}(2M-1) = h(M-1)y_1(1) + h(2M-1)y_1(0) + \cdots + h(\alpha+1)y_1(-m+2)$$

By comparing Eqs. (2-27) with (2-19), it is clear that both sets are identical with the only exception that the input sequence is incremented by 1. Hence the filter structure as shown in Figure 2-7 can be used with the input $y_1(1)$ to the filter. Repeating the above procedure, all interpolation outputs $\hat{y}(0)$, $\hat{y}(1)$, ... can be obtained. To show the efficiency of the new algorithm, the number of multiplications required for the 9.6 and 16.0 Kb/s subband coders, summarized in Table 2-6, shows that only half the number of multiplies is needed as compared to that of the conventional technique. Furthermore, utilizing this method, the mulipli-cation necessary for both digital decimation and interpolation are roughly equivalent.

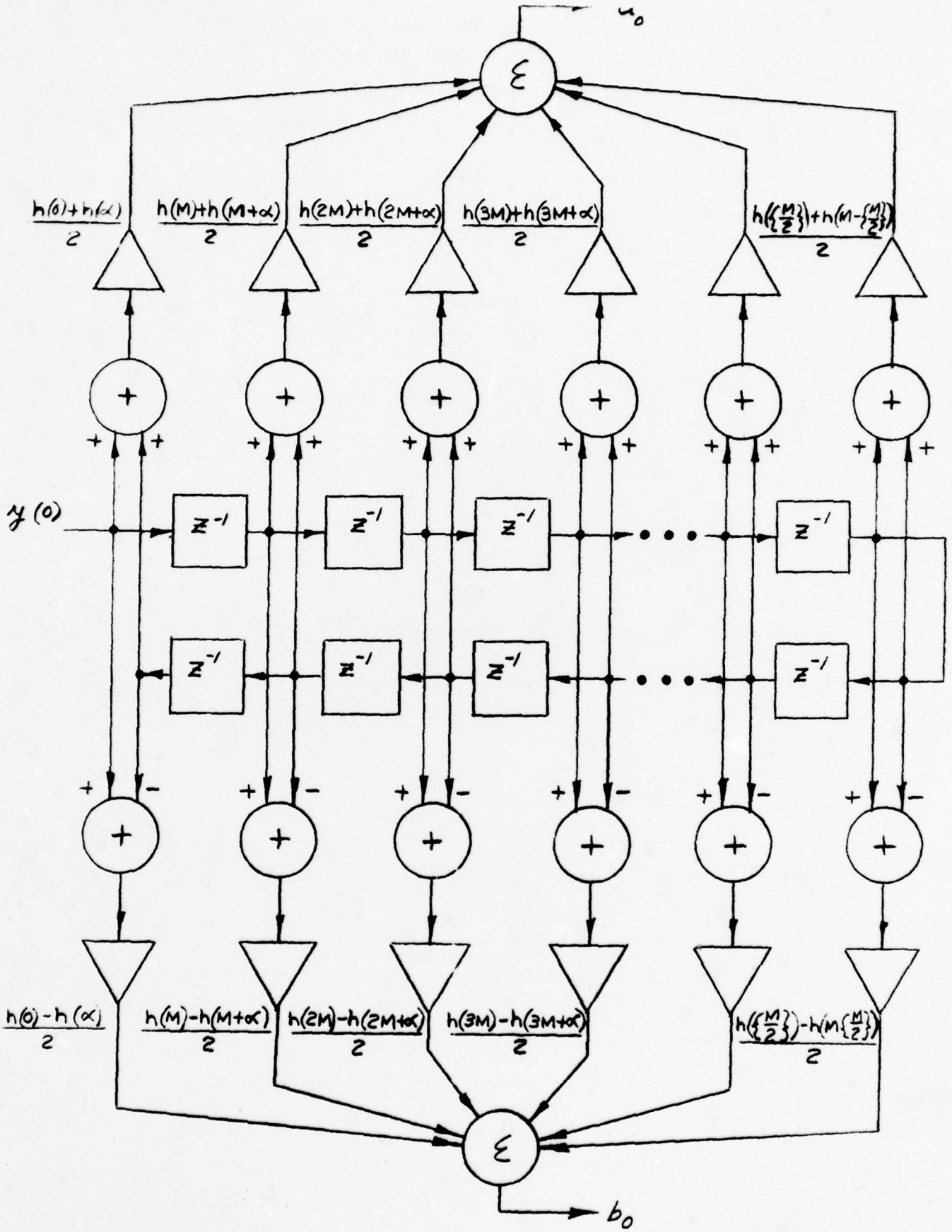


FIGURE 2-7 FILTER STRUCTURE FOR THE NOVEL INTERPOLATION SCHEME

SUBBAND #	# OF MULTIPLIES PER FRAME IN 9.6 KB/S SYSTEM	# OF MULTIPLIES PER FRAME IN 16.0 KB/S SYSTEM
1	900	330
2	1800	540
3	2020	900
4	3600	1800
5		1800
TOTAL	8320	5370

TABLE 2-6 NUMBER OF MULTIPLIES PER FRAME FOR DIGITAL INTERPOLATION WITH NEW TECHNIQUE

2.4 Alternative Subband Coding Techniques

From previous sections, it has been shown that subband coder is a conceptionally simple speech encoding scheme which deals mainly with the partitioning of input speech band into different frequency bands. As detailed in Section 2.2, one approach is to make use of finite-impulse response (FIR) filters to perform the bandpass filtering operation. However, to achieve the necessary bandpass characteristics, it is shown that large order filters are generally needed and this increases the processing burden. Moreover, the combined frequency response of the filter bank may not be desirable as a result of the spectral gaps as depicted in Figures 2-3 and 2-4. Hence, further improvements on performance or simplifications on the implementation of subband coders hinge on the utilization of other bandpass filters.

2.4.1 The Recursive Filter Approach

Since low order recursive filters, somtimes referred to as infinite-impulse response (IIR) filters, are known to yield magnitude response comparable to that of long FIR ones, it would seem feasible to use IIR filters in performing the bandpass filtering procedures in the subband coder. Unfortunately, nonuniform group delay due to the nonlinearity of IIR filter's phase response is rather undesirable for this application. Furthermore, the savings on computations (mainly multiplications) realized through the use of a short filter length is minimal because of IIR filter's inefficiency in performing digital interpolations and decimations.

To illustrate the latter statement, the use of an IIR filter to downsample input samples is considered. If the Z-transform of the IIR filter is:

$$H(z) = \frac{\sum_{i=0}^{N'-1} a_i z^{-i}}{\sum_{j=0}^{N'-1} b_j z^{-j}} \qquad (2\text{-}28)$$

where N' is defined as the order of the filter ($N' < N$), the relationship between the input and the filtered output sample $y(n)$ as depicted in Figure 2-2(a) is:

$$y(n) = -\sum_{j=1}^{N'-1} b_j\, y(n-j) + \sum_{i=0}^{N'-1} a_i\, x(n-i) \qquad (2\text{-}29)$$

Then the downsampled version of $y(n)$ is given by:

$$y_1(n) = y(Mn) = -\sum_{j=1}^{N'-1} b_j\, y(Mn-j) + \sum_{i=0}^{N'-1} a_i\, x(Mn-i) \qquad (2\text{-}30)$$

where M is the decimation ratio.

It is clear from Eq. (2-30) that in order to obtain $y_1(n)$, $2N'-1$ multiplications have to be performed for every $y(n)$. So, in contrast to the use of FIR filters, the amount of multiplications required in this case does not decrease with M. The use of IIR filters in subband coding is only advantageous over FIR ones if:

$$2N'-1 \geq N/2M$$

or

$$N' \geq \frac{1}{2}\left(\frac{N}{2M}+1\right) \qquad (2\text{-}31)$$

As the value of M increases, Eq. (2-31) is less likely to be true and this is one of the reasons that IIR filters may not be applicable for subband coding of speech signals.*

***Note: In a recent paper, the design of a special class of IIR filters is given which can perform efficient digital decimation [16]. However, the nonlinear phase characteristic of IIR filters is still undesirable for subband coding.**

2.4.2 The Quadrature Mirror Filter Approach

A better approach to subband coding is to make use of quadrature mirror filters (QMF) which band-splits/reconstructs the input waveform via decimation/interpolation methods [7]. It will be shown later than the use of QMF design equations will achieve perfect splitting/reconstruction without large-order filters.

For explanatory purposes, consider the ideal splitting/reconstruction process described in Figure 2-8. For this system, the following definitions apply:

a. $x(n)$ is a Nyquist band-limited signal with z-transform $X(z)$.

b. h_1 is the impulse response of the low-pass filter and the z-transform of which is $H_1(z)$.

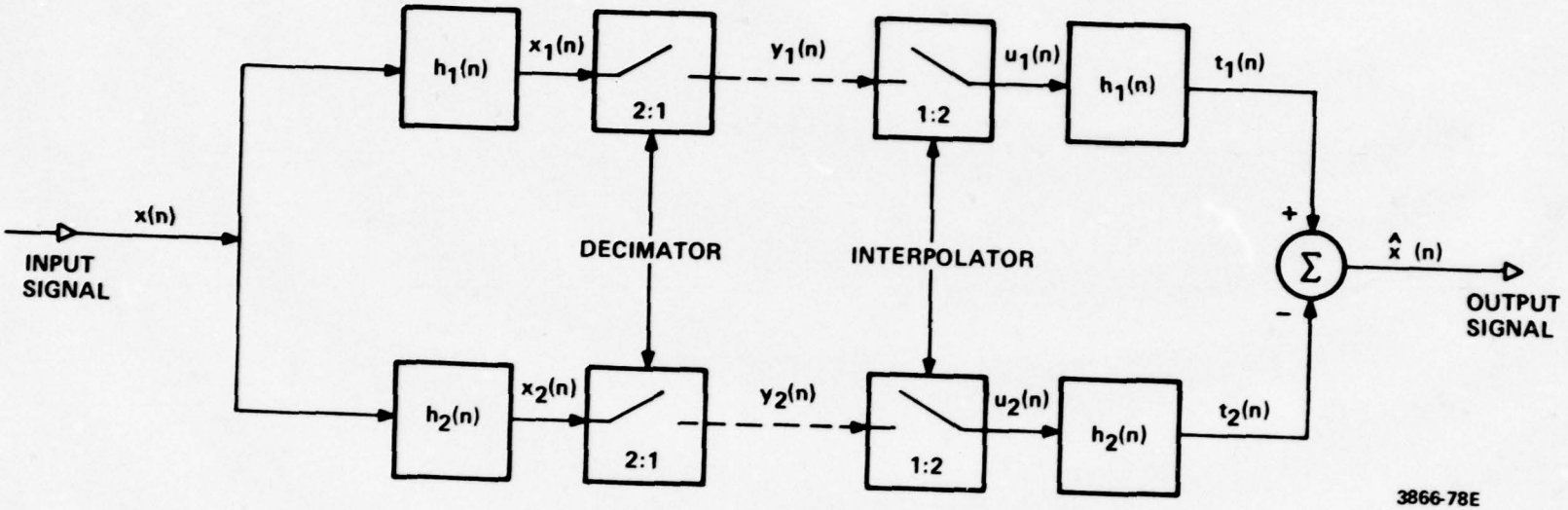
c. $h_2(n)$ is the impulse response of the high-pass filter the z-transform of which is $H_2(z)$.

d. $y_1(n)$ is a baseband equivalent low-pass signal with z-transform $Y_1(z)$.

e. $y_2(n)$ is a baseband equivalent high-pass signal with z-transform $Y_2(z)$.

The signal $x(n)$, is processed by filters $h_1(n)$ and $h_2(n)$ yielding the low-pass and high-pass equivalents, $x_1(n)$ and $x_2(n)$, of the input signal. As their spectra occupy half the Nyquist bandwidth of the original signal, the sampling rate in each band can be halved by decimating (ignoring) every second sample. For reconstruction, the signals $y_1(n)$ and $y_2(n)$ are interpolated, by inserting one zero-valued sample every other time, and then filtered respectivelyby $h_1(n)$ and $h_2(n)$ before being added, to give the signal $\hat{x}(n)$. The dashed lines, shown in Figure 2- 8, represent the data

FIGURE 2-8 **BAND-SPLITTING AND RECONSTRUCTION USING SINGLE-STAGE QMF**



passed to the communication channel(s) by the speech processing system.

In order to minimize $(\hat{x}(n) - x(n))$, certain restrictions on the filters, $h_1(n)$ and $h_2(n)$, must be met. We will derive these restrictions by contructing the transfer function of the QMF structure.

Using z-transform notation and referring to Figure 2-8, we may write the intermediary filtered output as

$$X_1(z) = H_1(z)X(z) \tag{2-32}$$

and

$$X_2(z) = H_2(z)X(z) \tag{2-33}$$

The transforms of the decimated signals, $y_1(n)$ and $y_2(n)$, and of the interpolated signals, $u_1(n)$ and $u_2(n)$, are given by:

$$Y_1(z) = \frac{1}{2}\left(X_1(\tilde{z}) + X_1(-\tilde{z})\right) \quad , \tilde{z} = z^{1/2} \tag{2-34}$$

$$Y_2(z) = \frac{1}{2}\left(X_2(\tilde{z}) + X_2(-\tilde{z})\right) \tag{2-35}$$

$$U_1(z) = Y_1(z^2) \tag{2-36}$$

$$U_2(z) = Y_2(z^2) \tag{2-37}$$

After the final filtering operating, the transforms of the reconstructed waveform components, $t_1(n)$ and $t_2(n)$, are given by

$$T_1(z) = H_1(z)\,U_1(z) \tag{2-38}$$

$$T_2(z) = H_2(z)\,U_2(z) \tag{2-39}$$

Using the relations expressed in (2-34) through (2-39), the z-transforms can be rewritten as

$$T_1(z) = \frac{1}{2}\left(H_1(z)X(z) + H_1(-z)X(-z)\right)H_1(z) \tag{2-40}$$

$$T_2(z) = -\frac{1}{2}\left(H_2(z)X(z) + H_2(-z)X(-z)\right)H_2(z) \tag{2-41}$$

The z-transform of the reconstructed waveform, $\hat{x}(n)$ is obtained by adding (2-40) and (2-41)

$$\hat{X}(z) = \frac{1}{2}\left(H_1^2(z) - H_2^2(z)\right)X(z) + \frac{1}{2}\left(H_1(-z)H_1(z) - H_2(-z)H_2(z)\right)X(-z) \tag{2-42}$$

If we assume that

$$H_2(z) = H_1(-z) \tag{2-43}$$

then the reconstructed waveform transform becomes

$$\hat{X}(z) = \frac{1}{2}\left(H_1^2(z) - H_1^2(-z)\right)X(z) \tag{2-44}$$

Evaluating z on the unit circle gives the Fourier transform of $\hat{X}(z)$

$$\hat{X}(e^{j\omega T}) = \frac{1}{2}\left(H_1^2(e^{j\omega T}) - H_1^2(e^{j(\omega + \frac{\omega_s}{2})T})\right)X(e^{j\omega T}) \tag{2-45}$$

For the case when $h_1(n)$ is an even, symmetrical FIR filter of order N, then it can be shown that (2-45) reduces to

$$\hat{X}(e^{j\omega T}) = \frac{1}{2}e^{-j(N-1)\omega T}X(e^{j\omega T}) \tag{2-46}$$

where $H_1^2(e^{jwt})$ exhibits an odd symmetric property about $^{w}s/4$ and the half-power point $H_1^2(^{w}s/4) = 0.5$.

The inverse transform yields a perfectly reconstructed signal (no frequency distortion) with a gain factor of 1/2 and delay of N-1 samplesas shown by

$$\hat{X}(n) = \frac{1}{2}x(n-N+1) \tag{2-47}$$

Therefore, we have shown that to guarantee perfect reconstruction of the original spectrum, the following filter constraints must be satisfied

$$h_1(n): \text{symmetrical, even order FIR} \quad (2\text{-}48)$$

$$H_2(z) = H_1(-z) \text{ or } h_2(n) = (-1)^n h_1(n) \; ; \; n = 0, 1, \ldots N-1 \quad (2\text{-}49)$$

$$H_1^2(z) + H_2^2(z) = 1 \quad (2\text{-}50)$$

2.4.2.1 Subband Coding via QMF Filters

Employing a procedure similar to the Hermann design algorithm for maximally flat FIR filters, a 60-tap filter which satisfies the QMF constraints as stated in Eqs. (2-48) to (2-50) is obtained [8,9]. Its impulse response $h_1(n)$ is tabulated in Appendix C and the corresponding magnitude response as depicted in Figure 2-9 shows that the filter has a flat passband extending from 0 - $f_s/4$ Hz. It is also characterized by a fast roll-off with the 3-db point located at $f_s/4$. From $h_1(n)$, its quadrature mirror counterpart $h_2(n)$ is generated using Eq. (2-49).

Starting with these two filters $h_1(n)$ and $h_2(n)$, a new form of subband coder is realized. As a matter of fact, a 3-stage tree structure as shown in Figure 2-10 is exploited to split the input speech band into 8 subbands [10]. For an input signal of 3200 Hz bandwidth, subbands of 400 Hz are resulted. At the first stage of the transmitter, the QMF filters split the input into two bands; that is, 0-1600 Hz and 1600-3200 Hz. Then a downsampling procedure is utilized to reduce the number of samples by a half. At the second stage, the identical bandsplitting process is applied to each subband. Consequently, 2 more subbands are generated each of which has a 800 Hz bandwidth. At the end of the second stage, a total of 4 subbands is obtained. The method is repeated one more time and eight subbands spanning frequency range from 0 to 3200 Hz

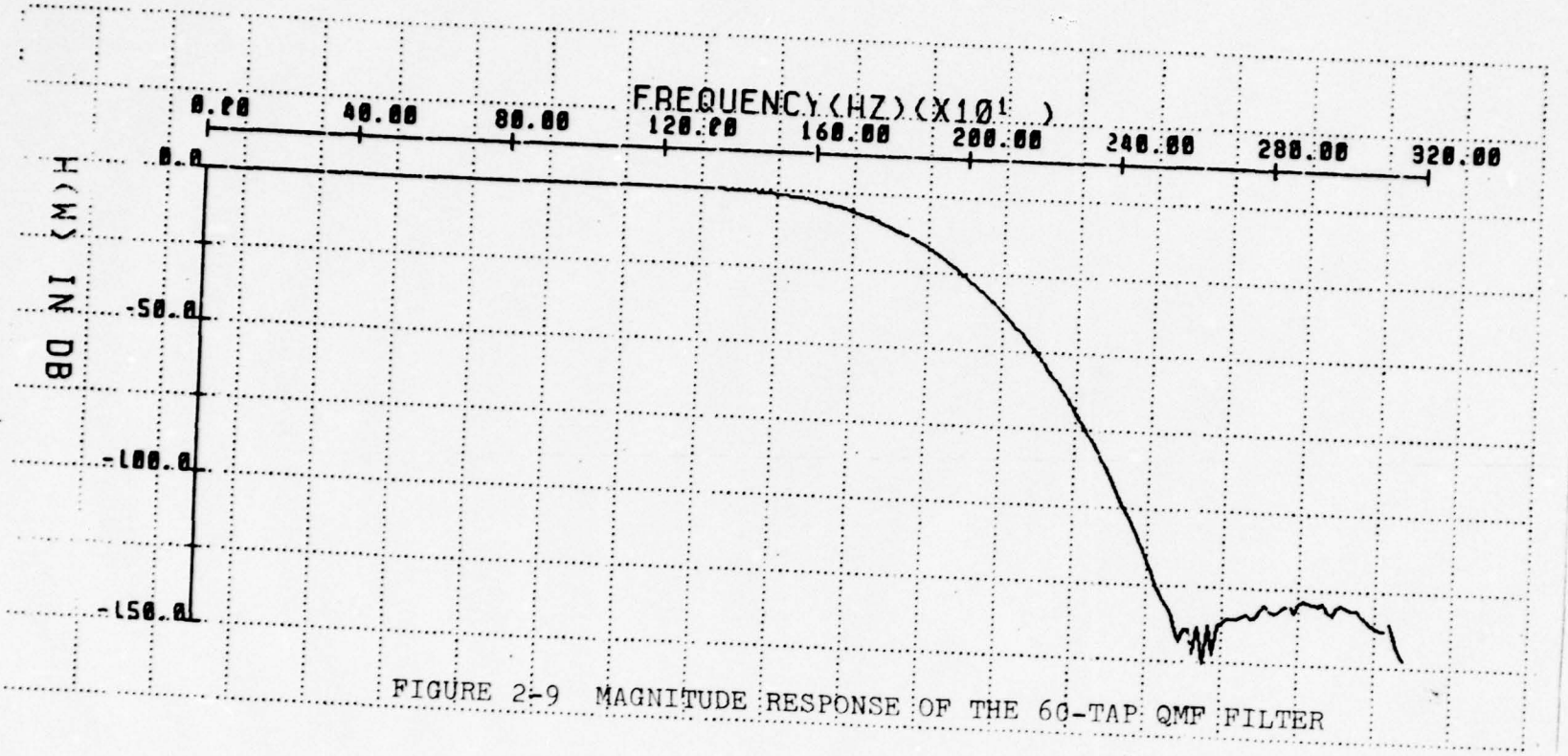


FIGURE 2-9 MAGNITUDE RESPONSE OF THE 60-TAP QMF FILTER

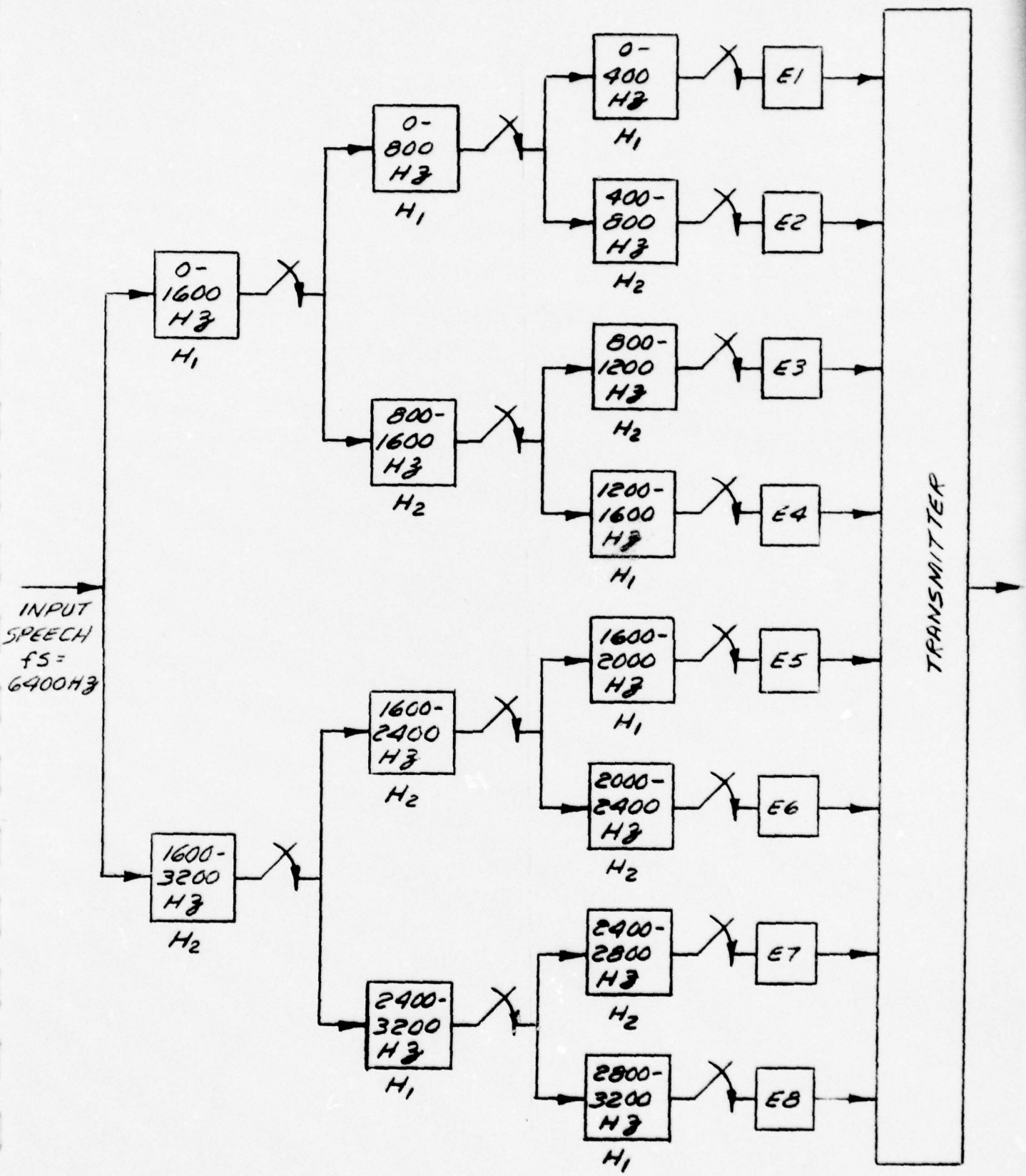


FIGURE 2-10(a) TRANSMITTER OF QMF SUBBAND CODER

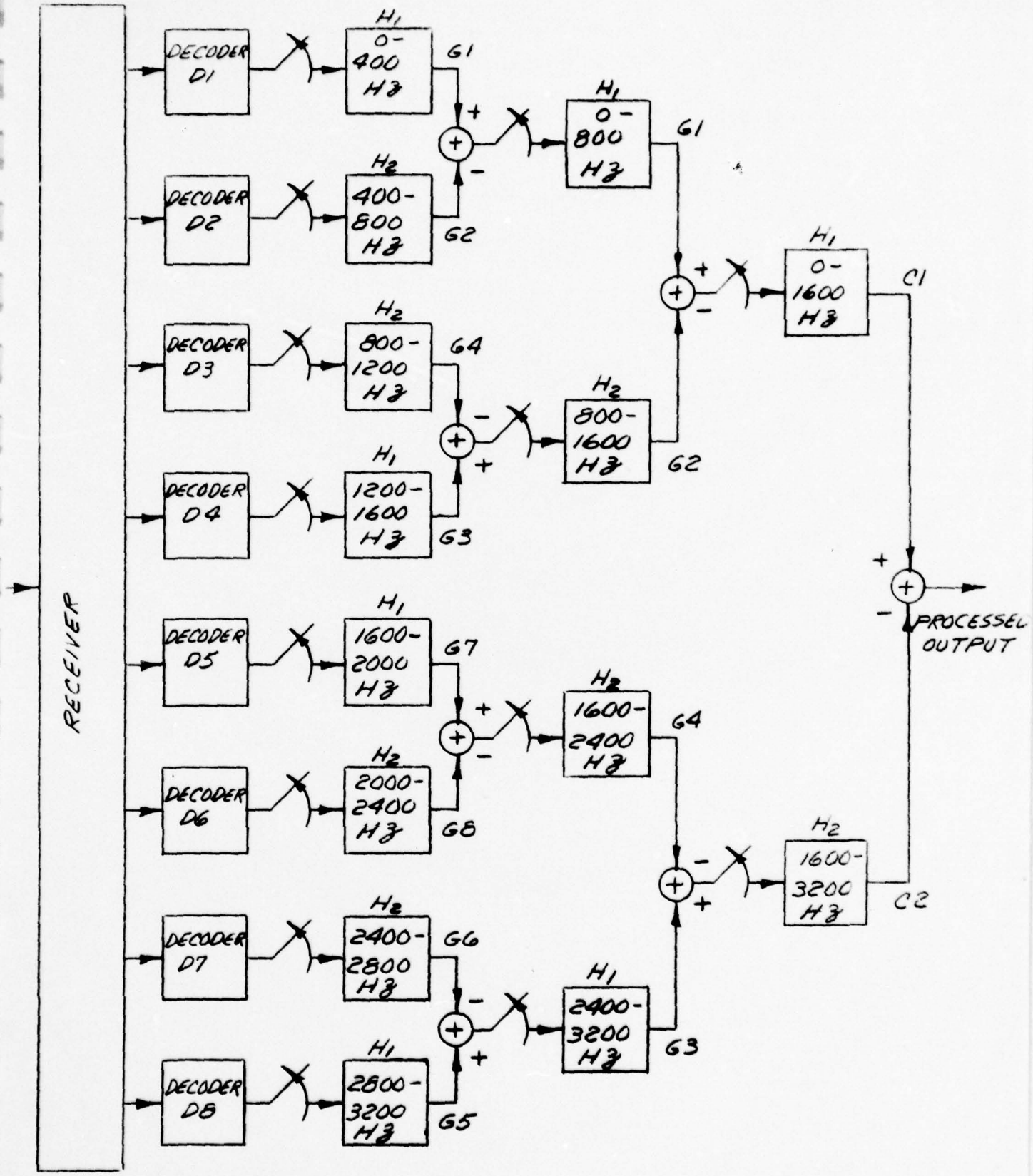


FIGURE 2-10(b) RECEIVER OF QMF SUBBAND CODER

are created. After performing the bandsplitting process, the subband signals are individually quantized for transmission. At the receiver, the reverse operation which includes up-sampling and bandpass filtering is performed. They are later recombined to form an estimate of the input signal. So, depending on the number of bits utilized in the quantization of the subbands, both the 9.6 and 16.0 Kb/s subband coders can be realized.

2.4.2.2 Adaptive Bit Allocations

One of the advantages of subband coding algorithms is the possibility of using the spectral densities of the suband signals and applying nonuniform quantization to each one. The conventional approach is to use a fixed-bit allocation rule where a pre-determined number of bits is assigned to the coding of each band. In the QMF system, a larger proportion of the available bits will be allocated to the quantization of the low bands in order to capture all pitch and formant information, whereas a smaller percentage of the bits will be utilized for the high bands. However, since the ratio of low-band and high-band energies fluctuates from frame to frame, fixed-bit allocation may not necessarily be the best strategy. Instead, an adaptive bit allocation scheme that dynamically alters the bit assignments depending on the signal energies of the eight bands seems more applicable [11].

For explanation purposes, let us assume that the average bit rate to quantize all subbands is $\bar{R}$ bits. In other words, if R_i is the actual number of bits allocated to the i^{th} subband, $\bar{R}$ is given by:

$$\bar{R} = \frac{1}{8} \sum_{i=1}^{8} R_i \qquad (2\text{-}51)$$

The number of bits needed to quantize the ith subband is given by [12].

$$R_i = \delta + \frac{1}{2}\log_2 \frac{\sigma_{x_i}^2}{\sigma_{q_{bi}}^2} \quad ; i=1,2,\cdots 8 \tag{2-52}$$

where $\sigma_{x_i}^2$, $\sigma_{q_i}^2$ denote the power of the i^{th} subband, and the corresponding quantization noise power, respectively, and δ is a correction term which depends on the performance of practical quantizers. If the dependence of R_i on δ is neglected, Equation (2-52) can be rewritten as:

$$\sigma_{q_{bi}}^2 = \sigma_{x_i}^2 \, 2^{-2R_i} \quad ; i=1,2,\cdots 8 \tag{2-53}$$

Defining the total quantization noise power E as

$$E = \sum_{i=1}^{8} \sigma_{q_{bi}}^2 \tag{2-54}$$

adaptive bit allocation calls for the selection of R_i for each subband such that E is minimized provided the constraint as given in Equation (2-51) is satisfied. This minimization procedure can be best performed with the help of Lagrange multipliers where a combined performance index I, defined as:

$$I = E + \lambda G \tag{2-55}$$

with λ given as the Lagrange multiplier and G the constraint function, is minimized. G is defined as:

$$G = \bar{R} - \frac{1}{8}\sum_{i=1}^{8} R_i \tag{2-56}$$

Differentiating I with respect to R_i and λ, and setting the resulting equations to zero, the following is obtained.

$$\sigma_{x_i}^2 \cdot 2^{-2R_i} + \lambda = 0 \qquad ; \; i = 1, 2, \cdots 8$$

$$\bar{R} = \frac{1}{8} \sum_{i=1}^{8} R_i \tag{2-57}$$

Combining Eq.(2-57) for i=1, 2, ... 8,

$$\sigma_{x_1}^2 \, \sigma_{x_2}^2 \cdots \sigma_{x_8}^2 \, 2^{-2(R_1 + R_2 + \cdots + R_8)} = \lambda^8 \tag{2-58}$$

Taking logarithm on both sides, substituting Equation (2-57) and eliminating λ, Equation (2-58) becomes

$$R_i = \bar{R} + \frac{1}{2} \log_2 \frac{\sigma_{x_i}^2}{(\sigma_{x_1}^2 \cdots \sigma_{x_8}^2)^{1/8}} \qquad i = 1, 2, \cdots 8 \tag{2-59}$$

Hence, bits to encode each subband can be adaptively allocated according to the corresponding signal power within each band. Furthermore, zero bits are assigned whenever R_i as obtained from Eq. (2-59) are negative. To prevent the allocation of all the bits in one subband, a maximum value is imposed on each R_i. Moreover, the frame powers for the eight subbands have to be transmitted so that the receiver can decode the bits via an identical scheme.

2.4.2.3 Discussion of Results

In order to evaluate the performance of subband coders using QMF filters, simulations of the system are programmed in FORTRAN. Specifications of the 9.6 and 16.0 Kb/s systems are summarized in Table 2-7. In this simulation, a 3-stage QMF filter structure as shown in Figure 2-10 is employed in the bandsplitting and reconstruction. With a filter length of 60 taps, the delay for the entire system is 7*(60-1) or 413 samples.

To examine the effect of the filters, the system is implemented without quantization and Figure 2-11 depicts the overall frequency response of the filter structure. In contrast to the integer-band sampling technique, the QMF filters span the entire speech band with no significant gaps in the spectrum and this results in the "smoother" quality of the processed speech.

To further improve the quality, the adaptive bit allocation algorithm as discussed in Section 2.4.2.2 is also incorporated. In order to prevent the allocation of too many bits on a single subband, a maximum number of 5 bits is allowed. Moreover, for bands with zeroth bit assignment, low level random noise is inserted to eliminate the "low-pass" effect[13]. Though the resulting speech is slightly noisier, the added noise seems to fill the spectral gaps created by quantization of some subbands with no bits and the overall quality of the processed speech is improved.

Figures 2-12 and 2-13 show frame-to-frame signal-to-noise plots for the 9.6 and 16.0 Kbps QMF subband coder. The result indicates that the QMF systems yield relatively high signal-to-noise ratios. Also, the coders seem to perform the best when the input power is highest. This is not surprising since after the bandpass filtering, the subband with the largest average amplitude is captured with the most bits using the adaptive bit allocation

Sampling Rate = 6.4 KHz
Frame Length = 144 samples
No. of Frames/Sec. = 44.4444
Frame Time = 22.5 msec

For the 16.0 Kb/s system: total no. of bits per frame to quantize 8 subbands = 324; Overhead bits per frame = 36

For the 9.6 Kb/s system: total no. of bits per frame to quantize 8 subbands = 180; Overhead bits per frame = 36

SUBBAND #	FREQUENCY RANGE IN HZ	DOWNSAMPLING (INTERPOLATION) RATIO FOR ALL 3 STAGES	QUANTIZER BIT ALLOCATIONS
1	0- 400	8	adaptive
2	400- 800	8	adaptive
3	800-1200	8	adaptive
4	1200-1600	8	adaptive
5	1600-2000	8	adaptive
6	2000-2400	8	adaptive
7	2400-2800	8	adaptive
8	2800-3200	8	adaptive

TABLE 2-7 SPECIFICATIONS OF 9.6 AND 16.0 KB/S QMF SUBBAND CODERS

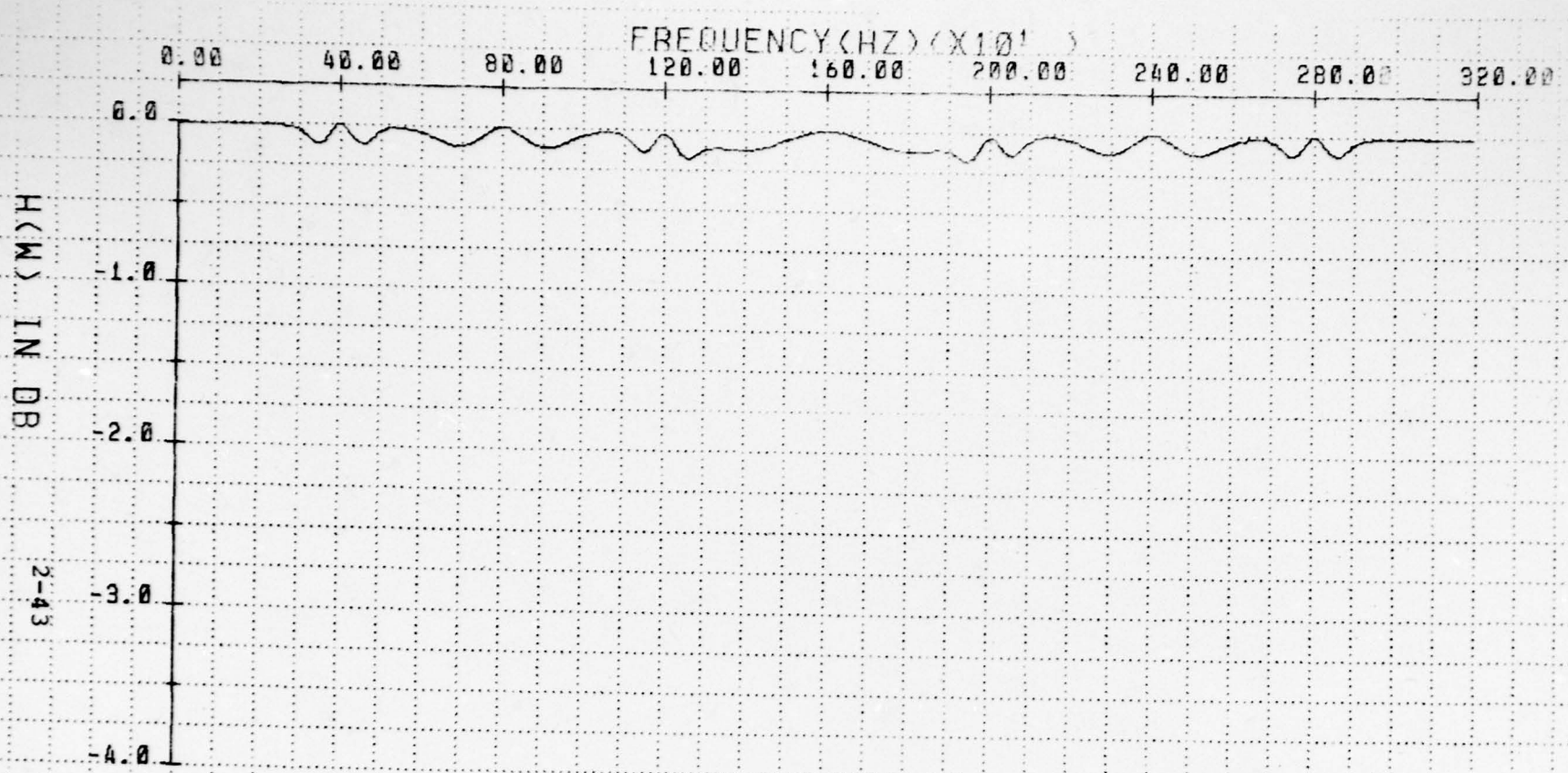


FIGURE 2-11 SPECTRAL RESPONSE 60 TAP QMF SUBBAND CODER

rule. Informal listening tests conducted on processed sentences also show that the speech quality of QMF subband coder is improved over that of the integer-band sampling technique. At 9.6 Kb/s, the system tends to produce slightly reverberent speech and however at 16 Kb/s, high quality speech is yielded.

FIGURE 2-12 A PLOT OF SIGNAL-TO-NOISE RATIO IN DB VERSUS FRAME NUMBER FOR THE 9.6 KB/S QMF SUBBAND CODER(CUMULATIVE SNR=15.90DB)

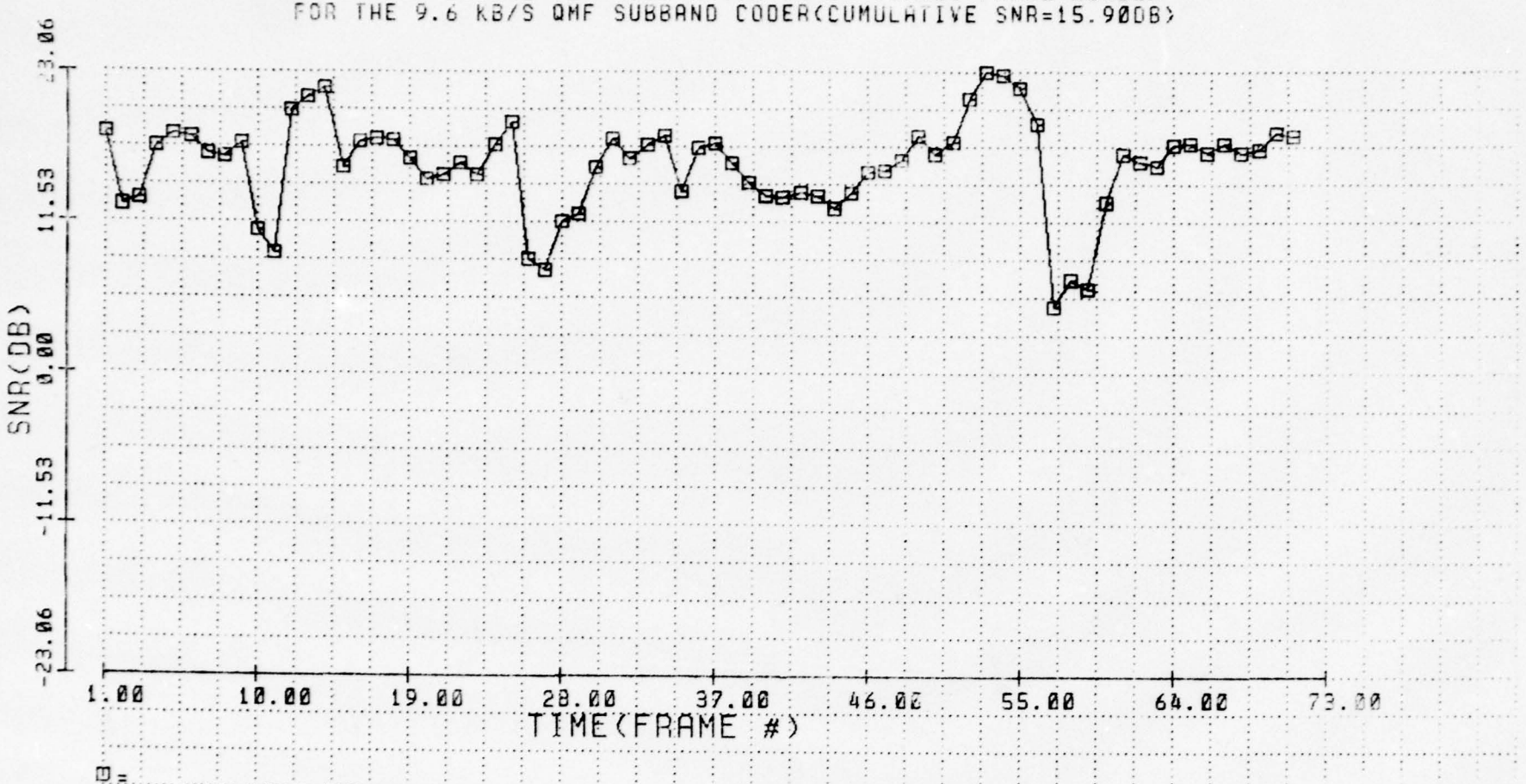
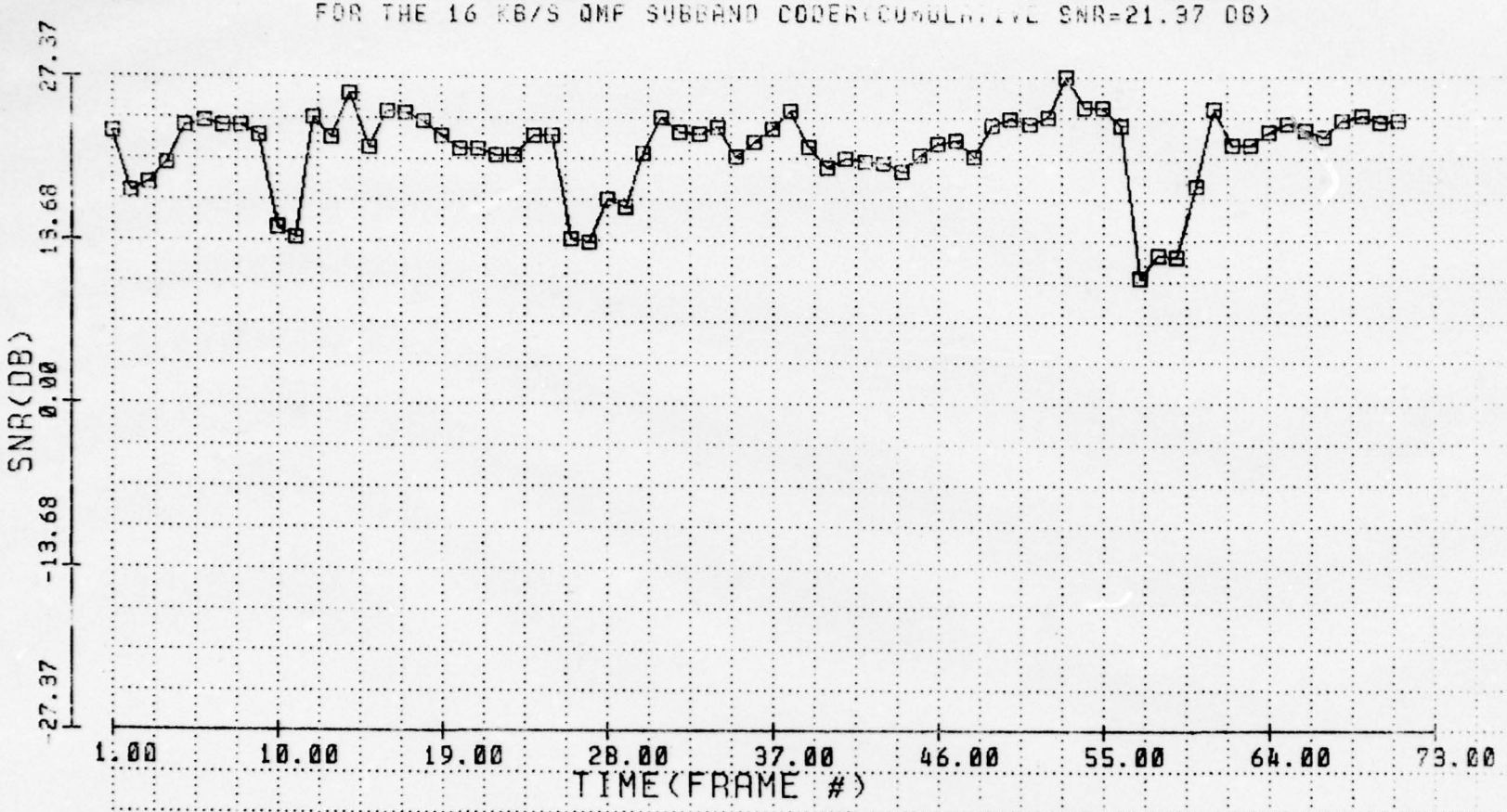


FIGURE 2-13 A PLOT OF SIGNAL-TO-NOISE RATIO IN DB VERSUS FRAME NUMBER FOR THE 16 KB/S QMF SUBBAND CODER(CUMULATIVE SNR=21.37 DB)



SECTION III
The Multiplier-Accumulator Hardware

3.1 General Description

In this contract, GTE Sylvania developed a state-of-the-art multiplier-accumulator (MULACC) which greatly enhances the capability of the Sylvania Programmable Signal Processor (PSP) in various signal processing applications. In addition to linear convolutions, the hardware performs computations of auto-/cross-correlation of two arrays of numbers most effectively. The nucleus of the design is a high speed 64-pin multiplier-accumulator chip (TRW TDC 1010J) which multiplies two 16-bit numbers and accumulates a 35-bit product in 115 nsec. Moreover, to fully exploit its capability, two buffers of fast RAM memories (MOSTEK MK4118P) are included which feed data directly into the chip. Since the circuitry communicates with the PSP CPU through input/output buses, these memory buffers drastically reduce the passing of data between the PSP and MULACC. Furthermore, since MULACC is treated as a peripheral, no hardware changes are required on the PSP CPU and all existing programs are still operable on the modified machine. After starting MULACC, the PSP CPU is also free to perform other tasks and this represents a more efficient utilization of available processing time. As an indication of its speed, the MULACC is capable of accessing the data from the two buffers, multiplying two 16-bit numbers, and forming a 35-bit product accumulation in 208 nsec (or 2 PSP cycles).

The MULACC hardware as shown in Figure 3-1 consists of two PC cards. The first card (board CON) , located at slot 7 of the PSP nest, is responsible for interpreting outputs from the PSP CPU, and decoding them into MULACC instructions. In particular, outputs from channel 5 of the PSP are converted into MULACC instructions whereas outputs from channel 6 are treated as data. Then the correct calling sequence for programming MULACC becomes

1) OUT 5 INSTRUCTION
2) OUT 6 DATA

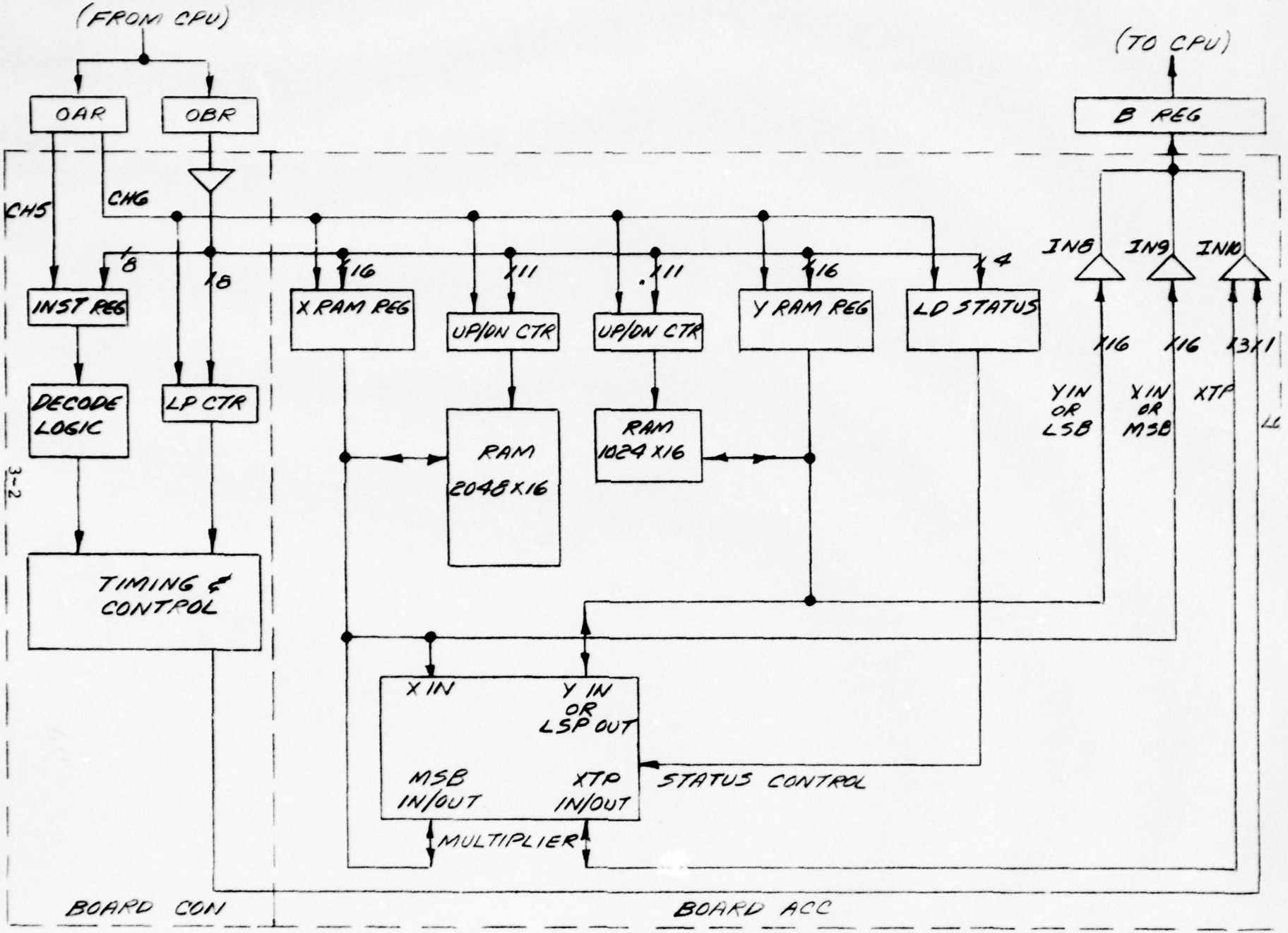


FIGURE 3-1 BLOCK DIAGRAM OF THE HIGH SPEED MULTIPLIER-ACCUMULATOR

After the first line of code, the instruction register of MULACC is loaded and the corresponding timing and control information are read from a programmable logic array (PLA). This is later used to setup the sequence of operations required by the instruction. From the second line of code, the data become operands of the corresponding memory registers and loopcounters.

The second PC card (board ACC) located at slot 5 of the PSP nest, contains the two buffer memories as denoted by X and Y, and the multiplier-accumulator chip. Memory address registers which are also present on board ACC can be programmed to point to any locations within the buffers. An up/down counter is utilized to increment/decrement the address registers automatically after each memory access. In the present design, 2K 16-bit data points can be pre-stored in X-buffer whereas 1K can be loaded into Y-buffer. After MULACC is started with setting of the loop counter, 32-bit product is first formed which is then added to the content of the TRW chip's output register. Upon the termination of the operation, a LOAD bit is set to a 1 and the 35-bit product is then transferred to the PSP input channels in three segments; namely, extended, most significant, and least significant products.

3.2 Programming of MULACC

As discussed in Section 3.1, procedures for programming MULACC include the outputting of an instruction code from channel 5 of the PSP, followed by an output of data from channel 6. From the first output, the instruction register of MULACC is loaded and the second output supplies the operand data. MULACC instructions, which include the loading of X and/or Y buffer addresses and memories, the selection of multiplier functions, and loading of multiplier loop counter are listed in Table 3-1.

As for multiplier functions, the hardware permits the choice of 12 functions; namely, unsigned magnitude/2's complement arithmetic, multiplication with/without rounding, multiplication with/without accumulation, multiplication with/without subtraction, increment/decrement of X-buffer pointers, and increment/decrement of Y-buffer pointers. Bit settings of the corresponding functions

TABLE 3-1 MULACC INSTRUCTIONS

INSTRUCTIONS	HEX VALUES	COMMENTS
Idle (IDLE)	0	An output of value 0 to channel 5 will set the MULACC to an idle state
Load X-Buffer Address (XADR)	1	Outputs of value 1 to channel 5, followed by the number XXXX to channel 6 will point to address XXXX of X-Buffer ($0 \leq$ XXXX $\leq$ 2047).
Load Y-Buffer Address (YADR)	2	Outputs of value 2 to channel 5 followed by the number YYYY to channel 6 will point to address YYYY of Y-Buffer ($0 \leq$ YYYY $\leq$ 1023).
Load Identical X-Buffer and Y-Buffer addresses (ADR)	3	Outputs of value 3 to channel 5, followed by the number XXXX to channel 6 will set both X and Y-Buffer address pointers to XXXX.
Load X-Buffer Memory (LDX)	4	Outputs of value 4 to channel 5, followed by the number DDDD to channel 6 will load data DDDD into the pointing X-Buffer address (see Note 1 in Table 3-4).
Load Y-Buffer Memory (LDY)	5	Outputs of value 5 to channel 5, followed by the number DDDD to channel 6 will load data DDDD into the pointing Y-Buffer address (see Note 1 in Table 3-4).
Load Identical Values into X and Y-Buffer Memories (LXAY)	6	Outputs of value 6 to channel 5, followed by the value DDDD to channel 6 will load data DDDD into the pointing X and Y-Buffer addresses (see Note 1 in Table 3-4).
Select Functions (TASK)	7	Outputs of value 7 to channel 5 followed by the value DDDD to channel 6 set up the MULACC to perform task DDDD. (Refer to table 3-2 for specific task definitions.)
Read Multiplier (READ)	8	An Output of value 8 to channel 5 followed by a waiting period of 4 cycles will enable the products presently residing at the MULACC to be read (see Note 2 and refer to Table 3-3 for specific reads).
Load Loop Counter & Start Multiplier (LOOP)	9	Ouputs of value 9 to channel 5 followed by the number (255-nnn) to channel 6 will set up the MULACC to multiply nnn times. The maximum number allowed is 255. Immediately after the setting of the loop counter, multiplication begins.
Load different addresses of X and Y-Buffers (XYADR)	A	This is a short cut to load different X and Y-Buffer addresses. Outputs of value A to channel 5 followed by a value of XXXX to channel 6 and a value of YYYY to channel 6 will set the X and Y address pointers to XXXX and YYYY, respectively.

are summarized in Table 3-2. To illustrate this, the selection of multiplication with accumulation function is considered. First, a value 7 is outputted from PSP channel 5 to MULACC which signifies the choice of functions. Then an output of $(33)_8$ from channel 6 sets up MULACC to perform 2's complement arithmetic, multiply (with no rounding) with accumulation, and increment both X and Y buffer pointers after each operation. Hence, by resetting the buffer pointers, the multiplier can be started to perform multiplications with accumulation of two arrays of numbers.

When the multiplier is finished, the load (RDY) bit of MULACC will be set to 1 and the results can be transferred. Since the PSP is a 16-bit machine, the 35-bit product is shipped via three input channels; namely, the extended product (XTP) through channel 10, the most significant product (MSP) through channel 9, and the least significant product (LSP) through channel 8. In addition, the RDY bit is multiplexed with the actual XTP during the transfer. So, after reading the XTP, the PSP has to perform a masking followed by a shift right once operation in order to obtain the correct XTP.

Besides the products, input channels 8 and 9 are also utilized to read the X and Y buffer memories. A tri-state device gating on the multiplier RDY bit is used to switch the outputs from the multiplier to that of the memories. In particular, when RDY = 0, input channels 8, 9 are connected to that of Y, X buffers, respectively. On the other hand, if RDY = 1, input channels 8, 9, 10 are hooked up to LSP, MSP, XTP. PSP instructions required to read products and memories are shown in Table 3-3. Also, special considerations on loading and reading of MULACC buffers, together with multiplier functions are discussed in Table 3-4.

To further illustrate the utility of MULACC, a PSP demonstration program that computes the following operation is detailed in Figure 3-2:

$$y(0) = \sum_{k=0}^{199} h(k)x(0-k) \qquad (3\text{-}1)$$

WAIT LOOP

TABLE 3-2 MULTIPLIER FUNCTIONS

	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
BIT SETTING = 0	DY	DX	NO RND	NO SUB	NO ACC	USM
BIT SETTING = 1	IY	IX	RND	SUB	ACC	TC

Symbol Definitions:

USM: Unsigned magnitude arithmetic (16-bit with no sign bit)

TC : 2's complement arithmetic (15-bit with 1 sign bit)

ACC: product accumulation (ACC = Product + ACC)

SUB: product subtraction (If bit 2=1 & bit 1=1, ACC = PRODUCT - ACC)

IX : increment X-buffer pointer

DX : decrement X-buffer pointer

IY : increment Y-buffer pointer

DY : decrement Y-buffer pointer

RND: rounding (see Note 3 on Table 3-4)

TABLE 3-3 INSTRUCTIONS TO READ BUFFER MEMORIES AND MULTIPLIER PRODUCTS

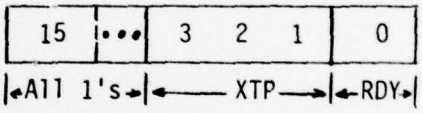
INSTRUCTIONS	PSP MNEMONICS	COMMENTS
To read contents of X-Buffer (XREAD)	INPA 9.	After setting up the X-Buffer address using XADR, its content can be read after a waiting period of 4 cycles
To read contents of Y-Buffer (YREAD)	INPA 8.	After setting up the Y-Buffer address using YADR, its content can be read after a waiting period of 4 cycles
To read the content of extended product (XTPR)	INPA 10.	Following either a load loop counter (LOOP) or read multiplier (READ) instruction, the extended product (XTP) can be accessed. Only the lower four bits are of interest and the upper 12 bits should be masked out. Its format is as follows:  where RDY=0: multiplication has not been completed RDY=1: multiplication has been completed The XTP is meaningful only if RDY=1
To read the content of the most significant product (MSPR)	INPA 9.	Following either a load loop counter (LOOP) or a read multiplier (READ) instruction, the most significant product (MSP) can be read. However the MSP value is valid only after the multiplication has been completed (i.e., RDY=1)
To read the content of the least significant product (LSPR)	INPA 8.	Following either a load loop counter (LOOP) or a read multiplier (READ) instruction, the least significant product (LSP) can be read. The LSP value is valid only after the multiplication has been completed (i.e., RDY=1)

TABLE 3-4 NOTES ON PROGRAMMING THE MULACC

1. Loading of X and/or Y-Buffer Memories

Since the loading of buffer memories takes more than two cycles, a 4-cycle wait period is needed to assure the success of writing consecutive memory locations.

2. Reading of Buffer Memories and Multiplier Products

The MULACC functions include the readings of buffer memories and the three multiplier products. In executing instructions that do not involve the multiplier, the product outputs are disenabled from the tri-state control and the bus is connected to the buffer memories which allows the reading of their contents. At this time, the multiplier ready bit is reset (RDY = 0). However, by outputting a read multiplier (READ) instruction on channel 5, the tri-state bus is switched to the multiplier outputs. This set the RDY bit to 1 and enables the reading of existing multiplier products.

In performing operations that require the multiplier, the RDY bit, originally set to 0, will change to a 1 immediately upon their completion. At this time, the three products will be ready to be transferred to the PSP using instructions shown in Table 3-2.

3. Multiply with Rounding

The TRW chip has a multiply with or with no rounding option. The multiply with rounding is performed by adding a 1 to bit 15 of the least significant product and the rounded result is obtained by reading the extended and most significant products. This option is generally not recommended for multiplication together with accumulation since it yields erroneous results. To further illustrate this, the multiplication with rounding and accumulation of two arrays of 0's shows a non-zero final value.

Block (1) of the program chooses the multiplier functions whereas Block (2) indicates the loading of the X and Y buffer memories. As shown in the figure, the starting addresses of the buffers are first set and then each MULACC buffer location can be individually loaded from PSP memories. By resetting the buffer pointers (X pointing to h(0), Y pointing to x(0)), the loop counter is initialized to be 200 and multiplication with accumulation is started as shown in Block (3). The multiplier RDY bit is constantly checked and results are transferred to PSP as illustrated in Block (4).

```
PAP-11 V31
                        NL=1,C
                              .TITLE DEMO.PAP
                              'THIS PROGRAM DEMONSTRATES THE USE OF MULACC TO PERFORM
                              'BANDPASS FILTERING
                              '
                              '   Y(0)=H(0)X(0)+H(1)X(-1)+...+H(199)X(-199)
                              '
                              '
                              '
                              'REGISTER FILE DEFINITION
            000000001         R1=1
                              '
                              '
                              'COMMAND DEFINTIONS
            00000B500         SETUP=132400                    'SAME OP CODE AS OUTM 5,0000
            000000660         DO=3140                         'SAME OP CODE AS OUTA 6
            000000820         LSPR=4040                       'SAME OP CODE AS INPA 8. :READ THE LEAST SIG. PART OF PRODUC
            000000920         MSPR=4440                       'SAME OP CODE AS INPA 9. :READ THE MOST SIG. PART OF PRODUCT
            000000A20         XTPR=5040                       'SAME OP CODE AS INPA 10.:READ THE EXTENDED PART OF PRODUCT
                              '
                              '
                              'MULACC FUNCTION DEFINITIONS
            000000001         XLN=01                          'MULTIPLY WITH NO ROUNDING
            000000002         ACC=02                          'ACCUMULATE
            000000010         IX=20                           'INCREMENT X BUFFER POINTER
            000000020         IY=40                           'INCREMENT Y BUFFER POINTER
                              '
                              '
            000000008         .PROG 10
                              '
                              '
0008  B500 0005               .INST           SETUP,  TASK                'SELECT MULACC FUNCTION
0009  7A43 0033                               LDAL    XLN!ACC!IX!IY       'FUNCTIONS ARE MULTIPLY WITH ACC,INCR. X AND INCR. Y
000A  0660 0000               .INST           DO
                              '
                              'LOADING OF BUFFERS X AND Y
000B  B500 0002               .INST           SETUP,  ADR                 'SET UP TO LOAD BOTH ADDRESSES OF BUFX AND BUFY
000C  7A43 0000                               LDAL    0                   'STARTING AT 0
000D  0660 0000               .INST           DO
                              '
                              '
                              'LOAD FILTER COEFFICIENTS INTO X BUFFER
000E  3A41 00C7                               LDRL    200.-1,R1           'SET UP COUNTER TO LOAD 200 COEFS
000F  B500 0003               .INST           SETUP,  LDX                 'SET UP INSTRUCTION TO LOAD X MEMORY
0010  CA21 0185  XLOAD$                       LDA     H ,R1               'PICK UP FILTER COEFS -
0011  0660 0000               .INST           DO                          'AND STORE IN BUFX
0012  1141 0010                               JDI     XLOAD,R1            'LOOP UNTIL ALL 200 COEFS ARE FINISHED
                              '
                              '
                              'LOAD INPUT SAMPLES INTO Y BUFFER
0013  3A41 018E                               LDRL    200.+199.-1,R1      'SET UP COUNTER TO LOAD 399 SAMPLES IN BUFFER Y
```

FIGURE 3-1 A PSP PROGRAM TO DEMONSTRATE THE UTILITY OF MULACC

```
XAP-11 V31                                DEMO.PAP
    53:  0014  B500 0004                .INST           SETUP,  LDY         'SET UP TO LOAD Y BUFFER
    54:  0015  CA21 000A  YLOAD$        .INST           LDA     MEMR,P1     'WITH MEMORY CONTENT FROM INPUT BUFFER
    55:  0016  0660 0000                .INST           DO
    56:  0017  1141 0015                                JDI     YLOAD,P1    'LOOP UNTIL ALL 399 SAMPLES ARE DONE
    57:                                 ;
    58:                                 ;
    59:                                 'MULTIPLY AND ACCUMULATE WITH MULACC
    60:  0018  B500 0000                .INST           SETUP,  XADR        'RESET THE X BUFFER POINTER TO BE AT 0
    61:  0019  7A43 0000                                LDAL    0
    62:  001A  0660 0000                .INST           DO
    63:  001B  B500 0001                .INST           SETUP,  YADR        'RESET THE Y BUFFER POINTER TO BE POINTING
    64:  001C  7A43 00B4                                LDAL    180.        'AT 180 OR INPT(0)
    65:  001D  0660 0000                .INST           DO
    66:  001E  B500 0006                .INST           SETUP,  LOOP        'RESET LOOP COUNTER TO PERFORM
    67:  001F  7A43 0037                                LDAL    255.-200.   '200 MULTIPLY AND ACCUMULATES
    68:  0020  0660 0000                ;.INST          DO
    69:                                 ;
    70:                                 ;
    71:  0021  0A20 0000  WAIT$         .INST           XTPR                'READ THE EXTENDED PRODUCT
    72:  0022  2020 0021                                JBR     0,WAIT      'IS READY BIT SET? CONTINUE LOOPING IF NO
    73:                                 ;
    74:                                 ;
    75:  0023  7B43 000F                                ANDL    15.         'IF READY, SAVE ONLY 4 LOWER BITS
    76:  0024  4C0B 0000                                SR1                 'SHIFT RIGHT ONCE TO GET RID OF READY BIT
    77:  0025  FF00 0007                                STA     XTP         'STORE IN MEMORY LOCATION XTP
    78:                                 ;
    79:                                 ;
    80:  0026  0820 0000                .INST           MSPR                'IF READY,READ THE MOST SIGNIFICANT PRODUCT
    81:  0027  FF00 0008                                STA     MSP         'STORE IN MEMORY LOCATION MSP
    82:                                 ;
    83:                                 ;
    84:  0028  0820 0000                .INST           LSPR                'IF READY, READ THE LEAST SIGNIFICANT PRODUCT
    85:  0029  FF00 0009                                STA     LSP         'STORE IN MEMORY LOCATION LSP
    86:                                 ;
    87:                                 ;
    88:                                 ;
    89:                                 'DATA MEMORY STARTS HERE.....
    90:        0000000000               .DATA   0
    91:  0000        0001  XADR$        1                                   'INSTRUCTION TO LOAD X BUFFER ADDRESS
    92:  0001        0002  YADR$        2                                   'INSTRUCTION TO LOAD Y BUFFER ADDRESS
    93:  0002        0003  ADR$         3                                   'LOAD IDENTICAL ADDRESS IN BUFX AND BUFY
    94:  0003        0004  LDX$         4                                   'LOAD DATA INTO BUFX
    95:  0004        0005  LDY$         5                                   'LOAD DATA INTO BUFY
    96:  0005        0007  TASK$        7                                   'SELECT FUNCTIONS
    97:  0006        0009  LOOP$        9.                                  'SET LOOP COUNTER AND START MULTIPLIER
    98:  0007        0000  XTP$         0                                   'LOCATION TO STORE EXTENDED PRODUCT
    99:  0008        0000  MSP$         0                                   'LOCATION TO STORE THE MOST SIG. PRODUCT
   100:  0009        0000  LSP$         0                                   'LOCATION TO STORE THE LEAST SIG. PRODUCT
   101:  000A  00000000C7  MEM$         .BLOCK 199.                         'FILTER MEMORIES X(-199),X(-198),...X(-1)
   102:  00D1  00000000B4  INPT$        .BLOCK 180.                         'INPUT BUFFER  X(0) X(1),... X(179)
   103:  0185  00000000C8  H$           ;.BLOCK 200.                        'FILTER COEFFICIENTS  H(199),H(198),...,H(0)
   104:                                 ;
   105:                                 ;
```

FIGURE 3-1 (CONT'D)

SECTION IV

Real-Time Software

4.1 Introduction

The real-time subband coder programs, operating at the data rates of 9.6 and 16.0 Kb/s, perform subband coding of speech signals via the integer-band sampling technique [3]. Exploiting a high speed multiplier-accumulator (MULACC), the 9.6 Kb/s program runs in a full-duplex mode whereas the 16.0 Kb/s program operates in half-duplex on the PSP. A summary of the real-time 9.6 and 16.0 Kb/s subband coder specifications is shown in Tables 2-1 and 2-2. The functions in the real-time program include speech/line side interrupt processing; receiver/transmitter synchronization; decimation/interpolation filtering; and quantization/dequantization of subband waveforms.

Under the speech side interrupt control, speech samples are brought into the PSP via the A/D converter and, concurrently, processed speech is outputted through the D/A. After subband filtering and encoding of the input signal, the binary bit stream is then transferred from one PSP to the other using the line side interrupt.

Within each frame of transmitted data, a synchronization (SYNC) bit is inserted at the same position of the frame. Through an extensive handshaking procedure, the receiver synchronizes with the transmitter by determining the position of the SYNC bit and eventually aligns the received data buffer to that of the transmitted one. In the real-time programs, the sync bit is set alternatively to a 1 or 0 to assure a more reliable synchronization between the two PSP's.

A major portion of the subband coder's processing load lies on the decimation and interpolation filtering in the main line code. At the transmitter, this involves the filtering of digitized speech samples using a bank of 4 or 5 finite-impulse-response (FIR) filters, each of which is 200-tap long, and the downsampling (decimating) of the output. These decimated subband signals are then quantized for transmission. At the receiver, the transmitted values are dequantized, up-sampled (interpolated), and filtered through the same bank of filters. The outputs are later combined to form an estimate of the input signal. The following sections describe in more detail the operation of each program function.

4.2 Speech/Line Side Interrupts

The speech side interrupt is responsible for transferring speech data into and out of the PSP's. Its interrupt rate is basically controlled by a programmable real-time counter ($000F_{16}$) and an I-O clock whose frequency is 11.52 MHz. Dividing the frequency of the clock by the setting of the counter yields the desirable sampling rate of the speech signal.

At every occurrence of a speech side interrupt, the program control is transferred to program counter = 0, which contains an instruction to jump to the speech side interrupt servicing routine where a speech sample is inputted to the speech buffer from the A/D converter and then a processed one is outputted to the D/A converter. The program utilizes a double buffering scheme on both the input and output buffers which allows the processing of a frame of data simultaneously with I/O accessing of the other. The speech side interrupt also sets flags and pointers indicating which buffers are being loaded or transmitted. Figure 4-1 describes in detail the operations of the speech side interrupt.

Similar to the speech side interrupt, the transfer of binary data from one PSP to the other is controlled by the line side interrupt whose rate is set by a second real-time counter ($000E_{16}$). At every occurrence of the interrupt, program control is sent to the line side interrupt servicing routine starting at program counter = 1 where transmission and reception of bit streams are performed. For the line side transmitter, the PSP hardware is set up in such a way that it takes two interrupts to generate 1 transmitter clock cycle (a high clock bit followed by a low one). At the same time, the identical data bit is outputted throughout the entire clock cycle. So, in order to achieve a particular data rate, the line side has to be interrupted at twice the data rate. For example, the line side interrupt rate has to be set at 32 KHz for 16 Kb/s transmission. As for the line side receiver, the routine inputs an 8-bit word on every 16th interrupt. Then each of these bits is unpeeled and a histogram of all receiver bits is compiled counting the number of times a 1 is received in each bit position. The histogram information is later used during the sync search. A flow chart description of the line side interrupt is depicted in Figure 4-2.

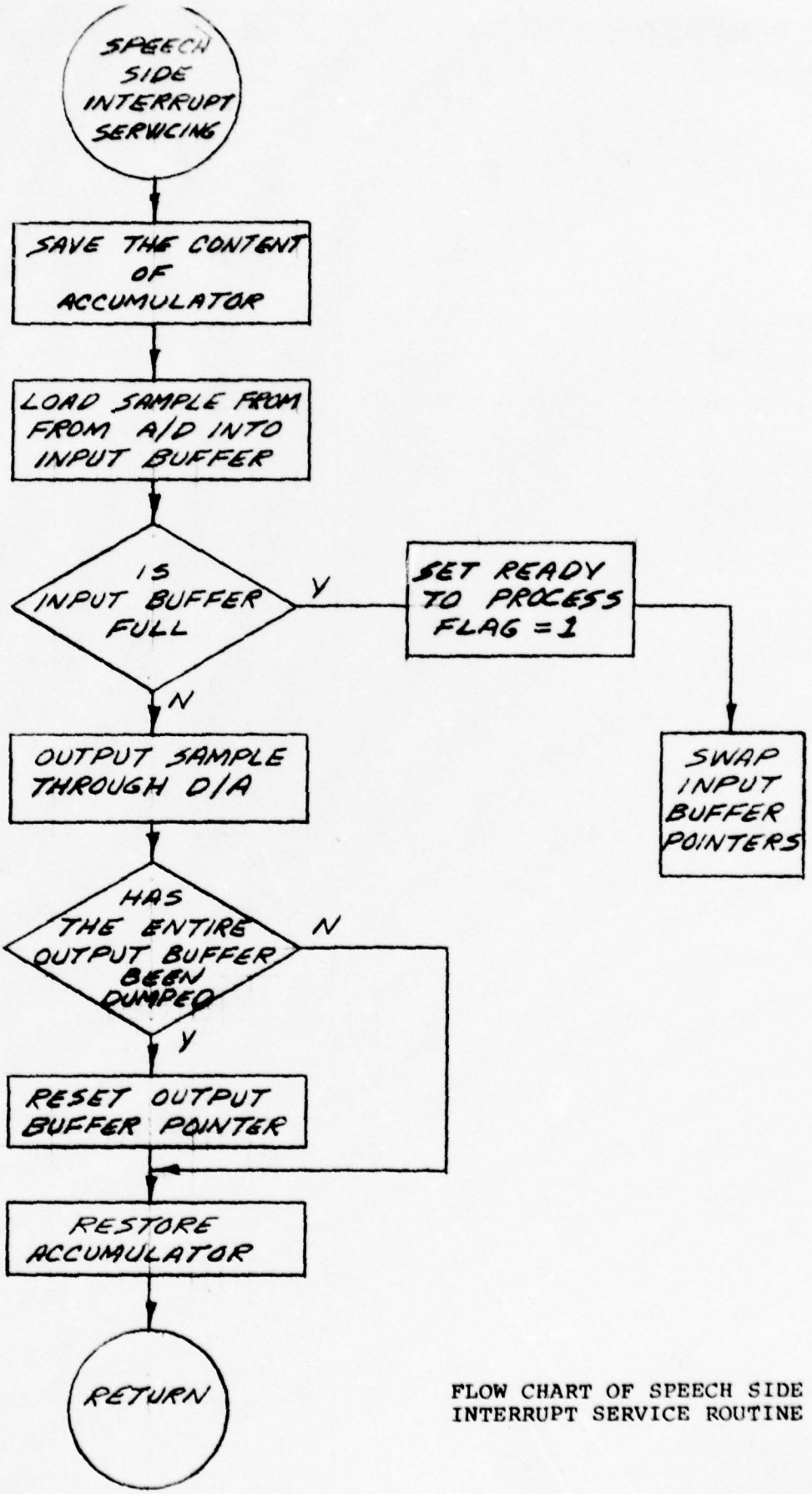


FIG 4-1

FLOW CHART OF SPEECH SIDE INTERRUPT SERVICE ROUTINE

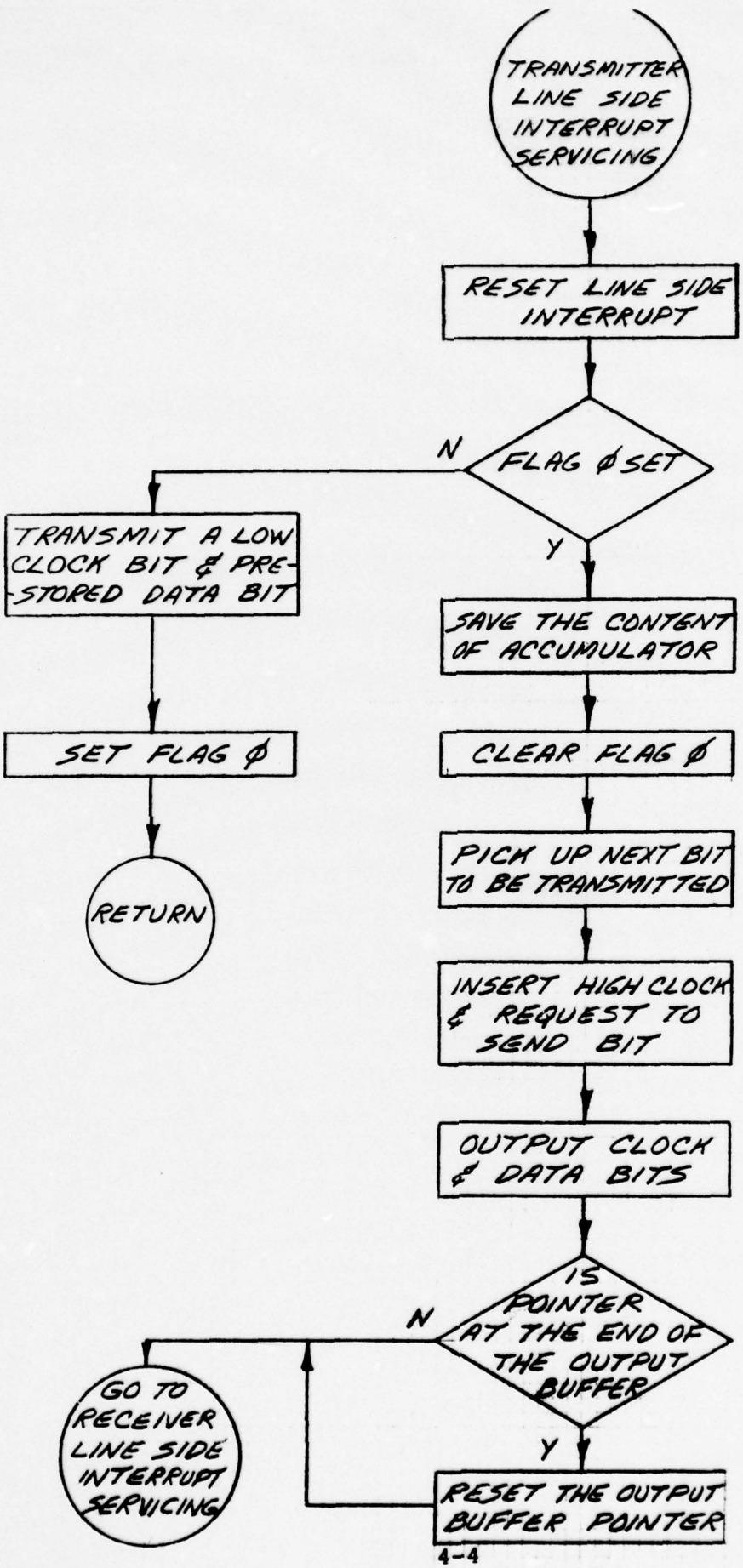


FIG 4-2(a)

FLOW CHART OF TRANSMITTER LINE SIDE INTERRUPT SERVICE ROUTINE

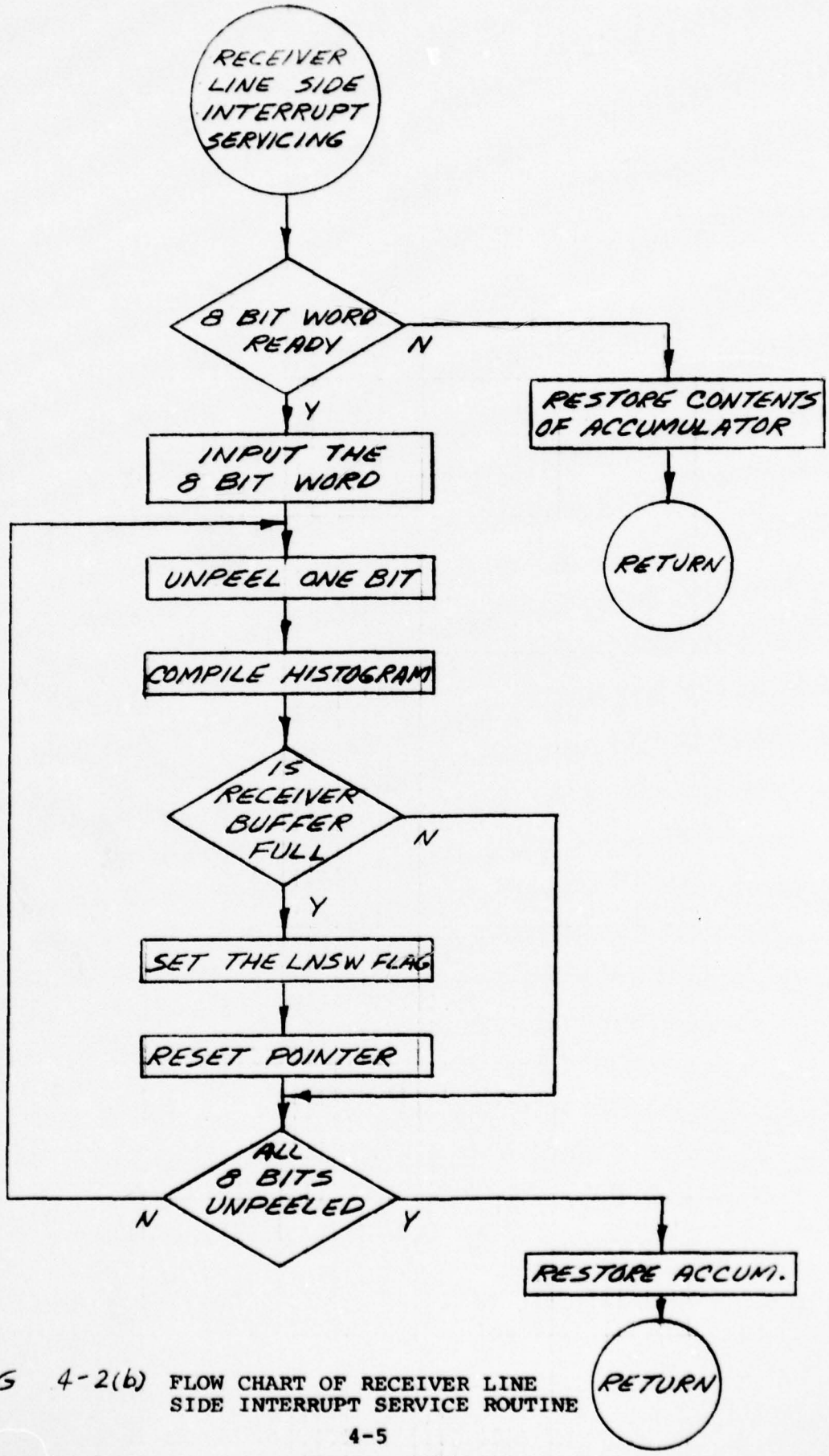


FIG 4-2(b) FLOW CHART OF RECEIVER LINE SIDE INTERRUPT SERVICE ROUTINE

of Y-buffer pointers. Bit settings of the corresponding functions

4.3 Initialization and Wait Loop

In the operation of the real-time program, the very first function is the initialization routine which sets most of the program variables prior to processing. All flags and storage buffers are cleared together with loading of the line and speech side real-time counters. Band-pass filter coefficients needed to perform decimation and interpolation filtering for all subbands are loaded into the X-buffer of the high speed multiplier-accumulator (MULACC). Figure 4-3 illustrates the storage of these coefficients. Also, quantizer/dequantizer parameters and index registers are loaded with their initial values. Upon exiting from the initialization routine, the speech and line side interrupts are enabled and program control is then transferred to the wait loop.

Figure 4-4 is a flow chart description of the wait loop. Since not all of the available processing time is used in the real-time subband coder programs, the wait loop consumes all the idle time left over. It also correctly sequences the transmitting and receiving operations by examining the transmitter and receiver ready flags and interrogates the front panel to see if reinitialization is desired.

4.4 Transmitter/Receiver Synchronization

After quantizing the subband signals, the transmitter data buffers in the real-time program are set up to have unused bits. In particular, there are 5 bits left over in the 9.6 Kb/s and 4 in the 16.0 Kb/s systems. One of these available bits is allocated in the programs for synchronization purposes. This bit, generally referred to as the SYNC bit, is set to a 1 or 0 every alternate frame. Utilizing a search procedure for this SYNC bit, the receiver is able to locate the beginning of the received buffer for proper decoding of the transmitted data. The sync search routine is performed on every 80th pass (or 675 msec) through the receiver codes in the 16.0 Kb/s system and on every 40th pass (or 750 msec) in the 9.6 Kb/s system. During the remaining times, the receiver line side compiles a histogram which tallies the times that a 1 has been received in each bit

Load different addresses of X and Y-Buffers (XYADR)	A	This is a short cut to load different X and Y-Buffer addresses. Outputs of value A to channel 5 followed by a value of XXXX to channel 6 and a value of YYYY to channel 6 will set the X and Y address pointers to XXXX and YYYY, respectively.

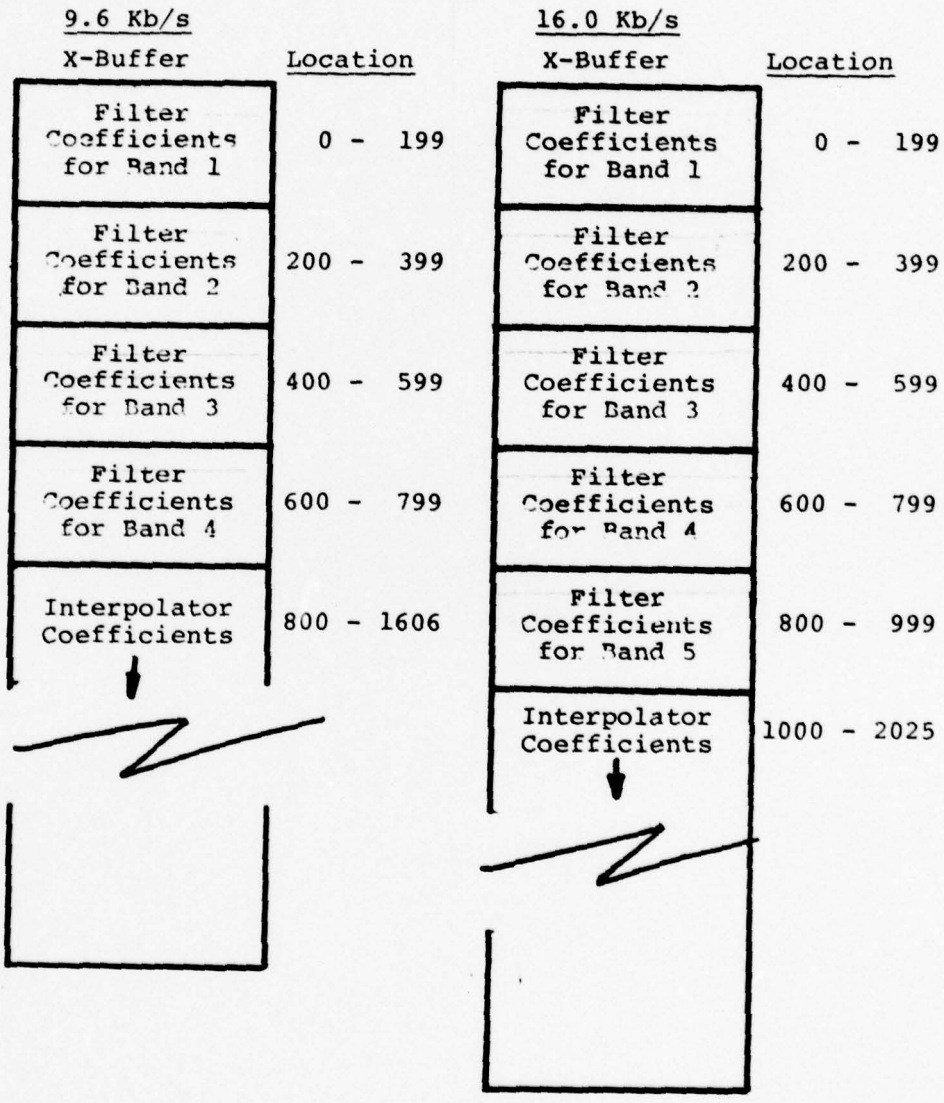


FIGURE 4-3
Storage of Filter Coefficients in MULACC

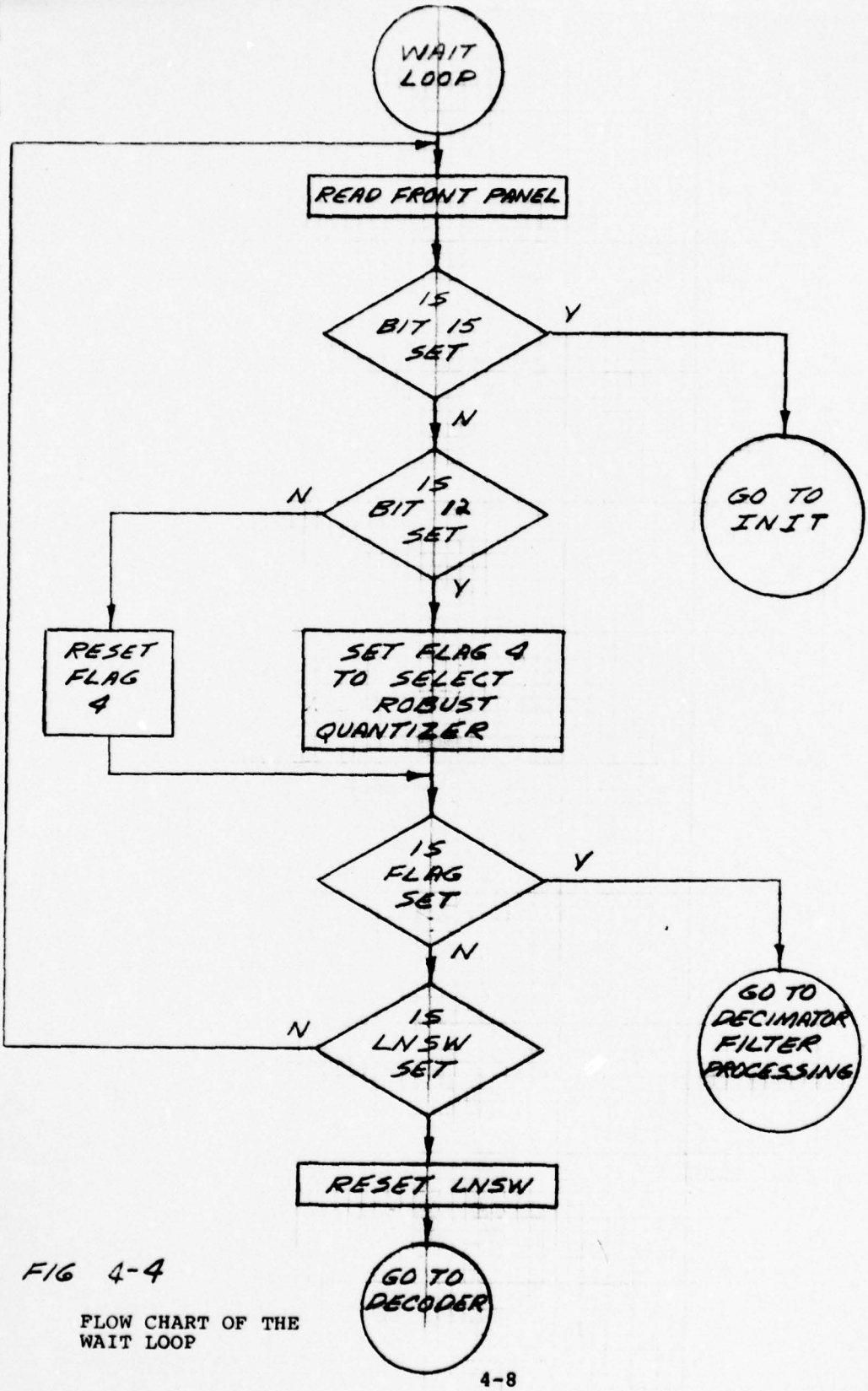


FIG 4-4

FLOW CHART OF THE WAIT LOOP

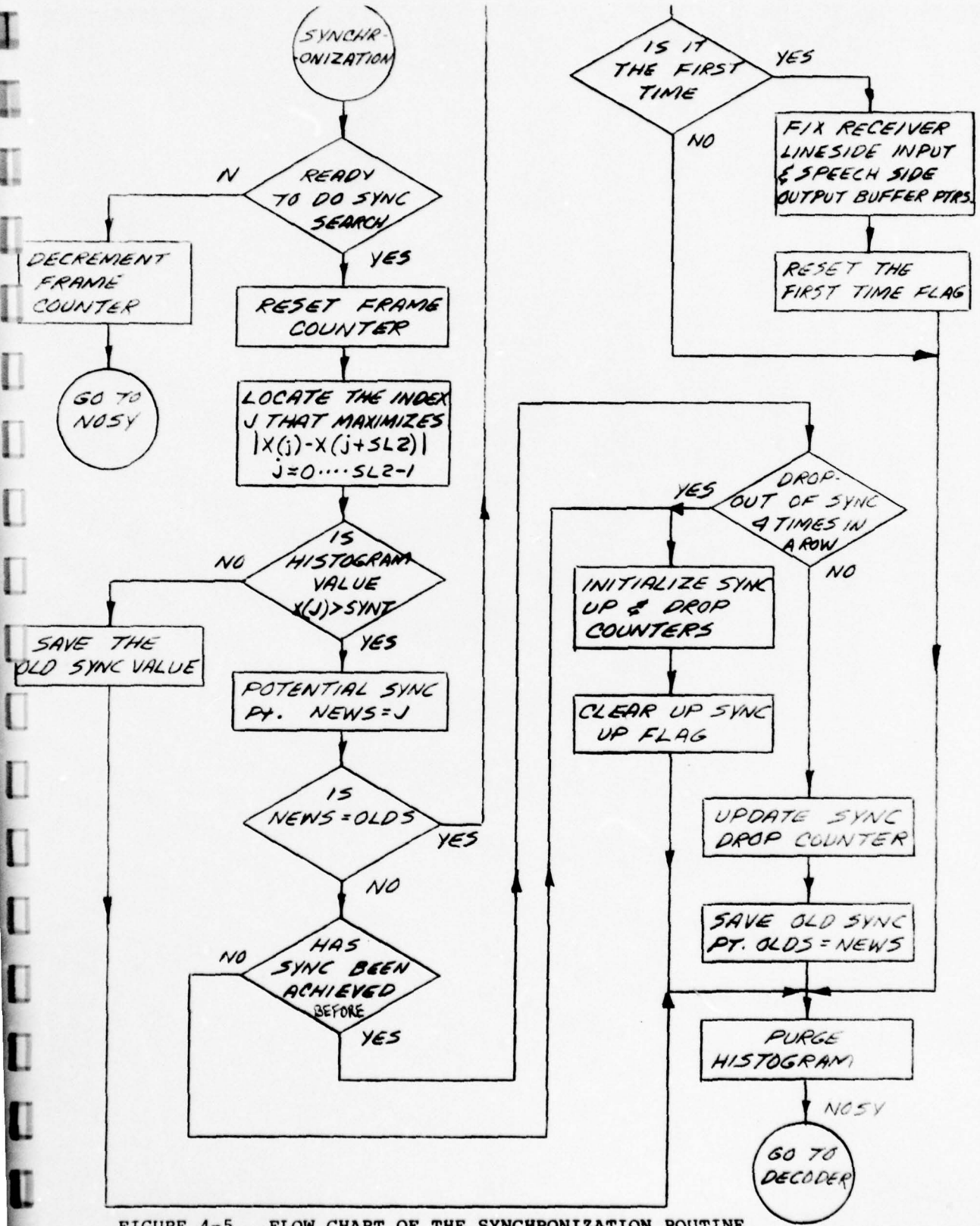


FIGURE 4-5 FLOW CHART OF THE SYNCHRONIZATION ROUTINE

position within the two data buffers. The sync position is then located by computing the absolute difference between the histogram values of the present bit position and that of the bit situated one frame apart. The bit position that corresponds to the largest difference is defined as the sync point. To assure that the correct sync point has indeed been found, sync search is repeated two more times. If the sync point is identical in all 3 cases, the true one is located. Otherwise, the sync search is resumed until three identical sync points are consecutively found. After achieving synchronization, the sync search is still performed. A sync drop counter is utilized to keep track of the failure of synchronization. As a matter of fact, if the two PSP's are out of sync five times, resynchronization is automatically repeated. Figure 4-5 is a flowchart description of the synchronization routine.

4.5 Decimation Filtering

Once the initialization and synchronization procedures have been completed, the PSP's are ready to process the input speech. As described before, the transmitter ready flag (FLAG) is constantly examined in the wait loop to determine when a frame of new speech data has been loaded. When FLAG is set, program control is transferred to the transmitter routine whose major functions include decimation filtering.

Upon entering the transmitter code, a sync pulse is outputted for timing purposes and FLAG is reset. Then the filter memories are updated together with the loading of a new frame of speech data.

The decimation filters decompose the incoming speech signal into four (in the 9.6 Kb/s system) or five (in the 16.0 Kb/s system) subbands. The integer band down-sampling method is used to perform the low-pass translation of each subband. The speech samples stored in the decimation filter buffers are loaded into the Y-array of MULACC as shown in Figure 4-6. It is only necessary to load this buffer once every frame.

After the X- and Y-buffers of MULACC are loaded, the following filtering operation is implemented for each subband:

illustrate this, the multiplication with rounding and accumulation of two arrays of 0's shows a non-zero final value.

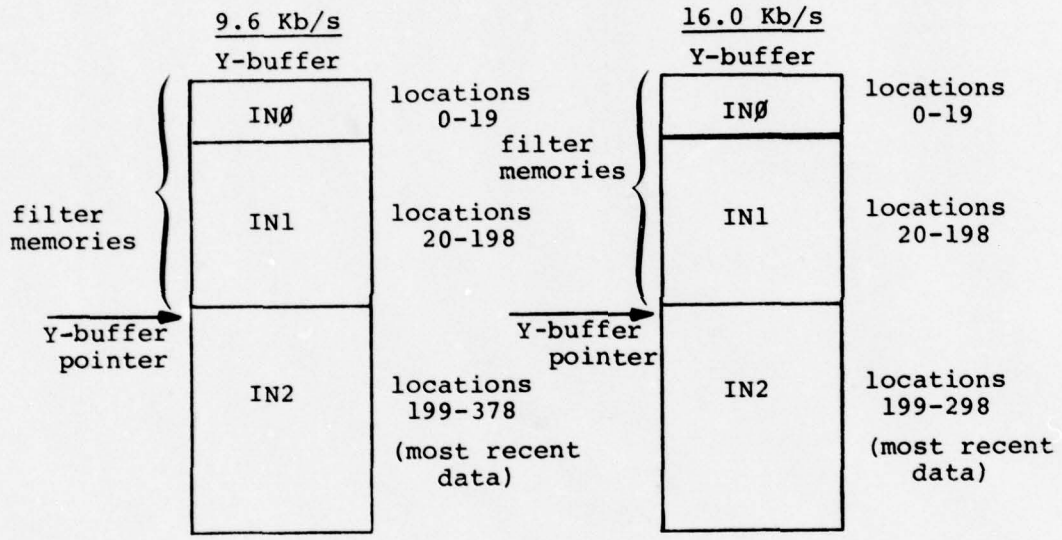


FIGURE 4-6 LOADING OF THE Y-BUFFER FOR DECIMATION FILTERING

$$y(nL) = \sum_{k=0}^{199} h(k)x(nL-k) \qquad (4\text{-}1)$$

where L is the decimation ratio, h(k) is the kth filter coefficient, x(·) is the input sample, and y(·) is the filtered output

Initially, MULACC is setup to perform multiply and accumulate with no rounding; increment X-buffer and decrement Y-buffer pointers. The X-buffer pointer is directed to the first filter coefficient, h(0), of band 1 and the Y-buffer one is set to point at x(0) which is stored at the 199th location of the Y-buffer. By setting the loop counter to be equal to 200, the filtering operation is started. After MULACC finishes, the ready bit in the extended product (XTP) is set and the result is transferred to PSP. Incrementing the Y-pointer by a factor of L and resetting the X-pointer to Ø computation of the second sample is performed. Since it takes MULACC 42 msec to compute each output point, the PSP is free to perform other functions. In the real-time programs, this time is used for the quantization of the previously computed sample, thereby hiding the amount of time needed for quantization.

When the point y(nL) is computed, the X and Y buffer pointers are reset to compute the point y((n+1)L). If all the points for the particular band have been calculated, program control is transferred to the filter loop of the next subband. Once the filtering for all bands is complete, the program leaves for the encoder routine. Figure 4-7 is a flow chart that describes the operation of the decimation filters for the 9.6 Kb/s system. The corresponding codes for the 16.0 Kb/s system are similar.

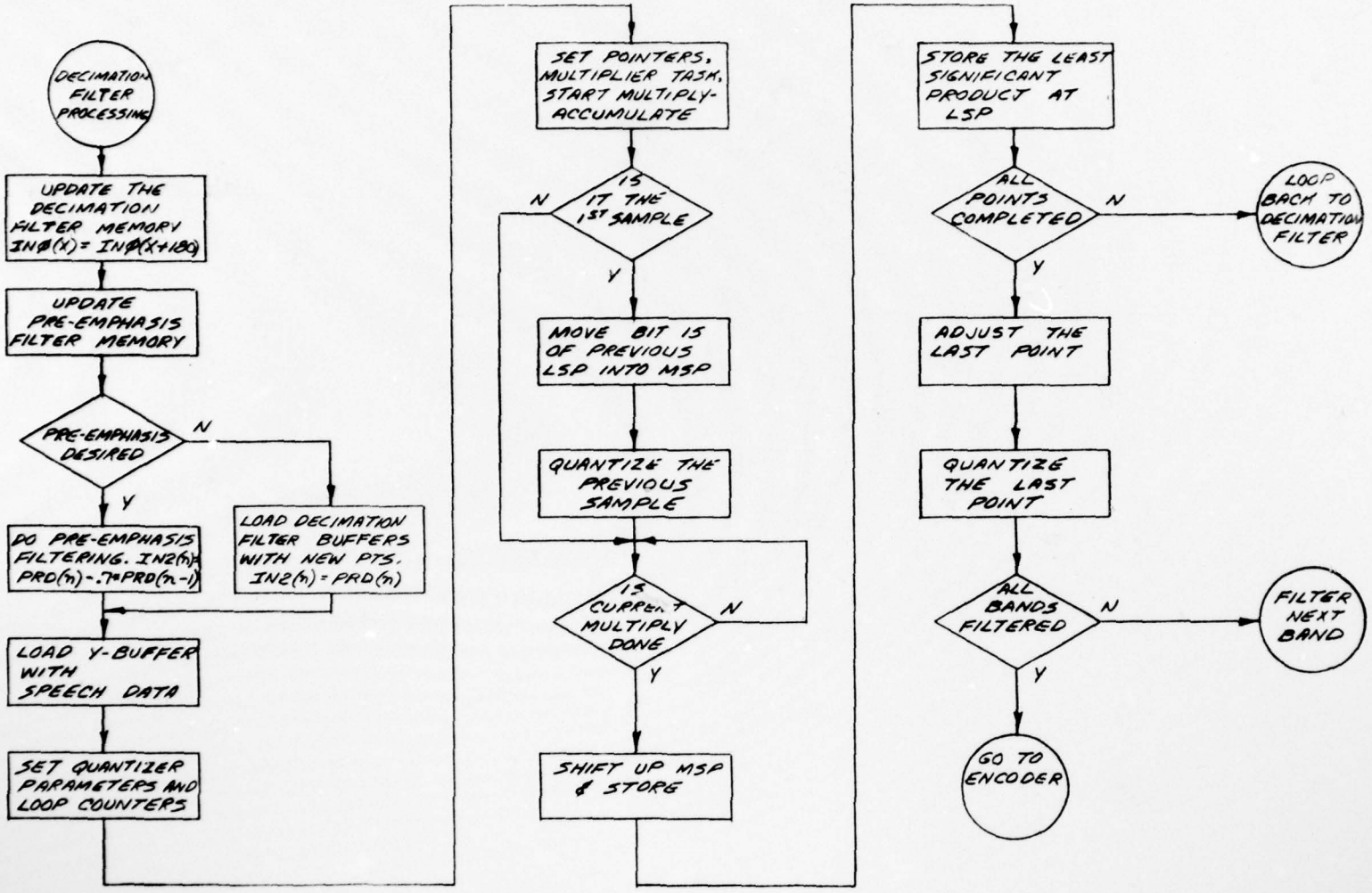


FIGURE 4-7 FLOW CHART OF DECIMATION FILTERING

4.6 Quantization and Encoding

After the band-pass filtering and decimation operations, the resulting subband signals are converted to binary bit streams using adaptive PCM (APCM) quantizers. Employing the adaptation strategy of one-word memory, the quantizer dynamically adjusts its step sizes according to the amplitude of the input signal. Maximum and minimum step sizes, adaptation and gain factors for the quantizers are summarized in Table 2-3. As described earlier, the quantization is done within the subband filtering loop in the program to minimize processing time.

Since the adaptive quantizer is known to perform poorly in the presence of channel errors, the robust algorithm as described in Appendix D is incorporated. In the actual implementation, the decaying factor β is chosen to be 0.96 and the function Δ^{β}, where Δ is the quantizer step-size, is estimated using a polynomial approximation procedure as described in Appendix E. To further reduce processing time, only a 3rd order polynomial is utilized. The flow-chart of an 8-level, robust quantizer is shown in Figure 4-8.

After quantizing the subband signals, the resulting bit stream is encoded and stored in the appropriate output buffer as depicted in Figure 4-9. The formats of the transmission data for both 9.6 and 16.0 Kb/s are shown in Figures 4-10(a) and (b). Upon finishing the encoding process, all transmitter functions have been completed. Then a sync pulse is outputted to denote the end of all transmitter functions and program control is transferred back to the wait loop.

4.7 Decoding and Dequantization

When the receiver ready flag (LNSW) is set equal to 1, the program will leave the wait loop and go to the receiver code. Decoding of the received data as shown in Figure 4.11 is the first function performed. Through this procedure, the receiver reconstructs code words from the data buffer each of which corresponds to a transmitted sample.

After the decoding, dequantizer routines are employed to convert the received code words to their corresponding values. A

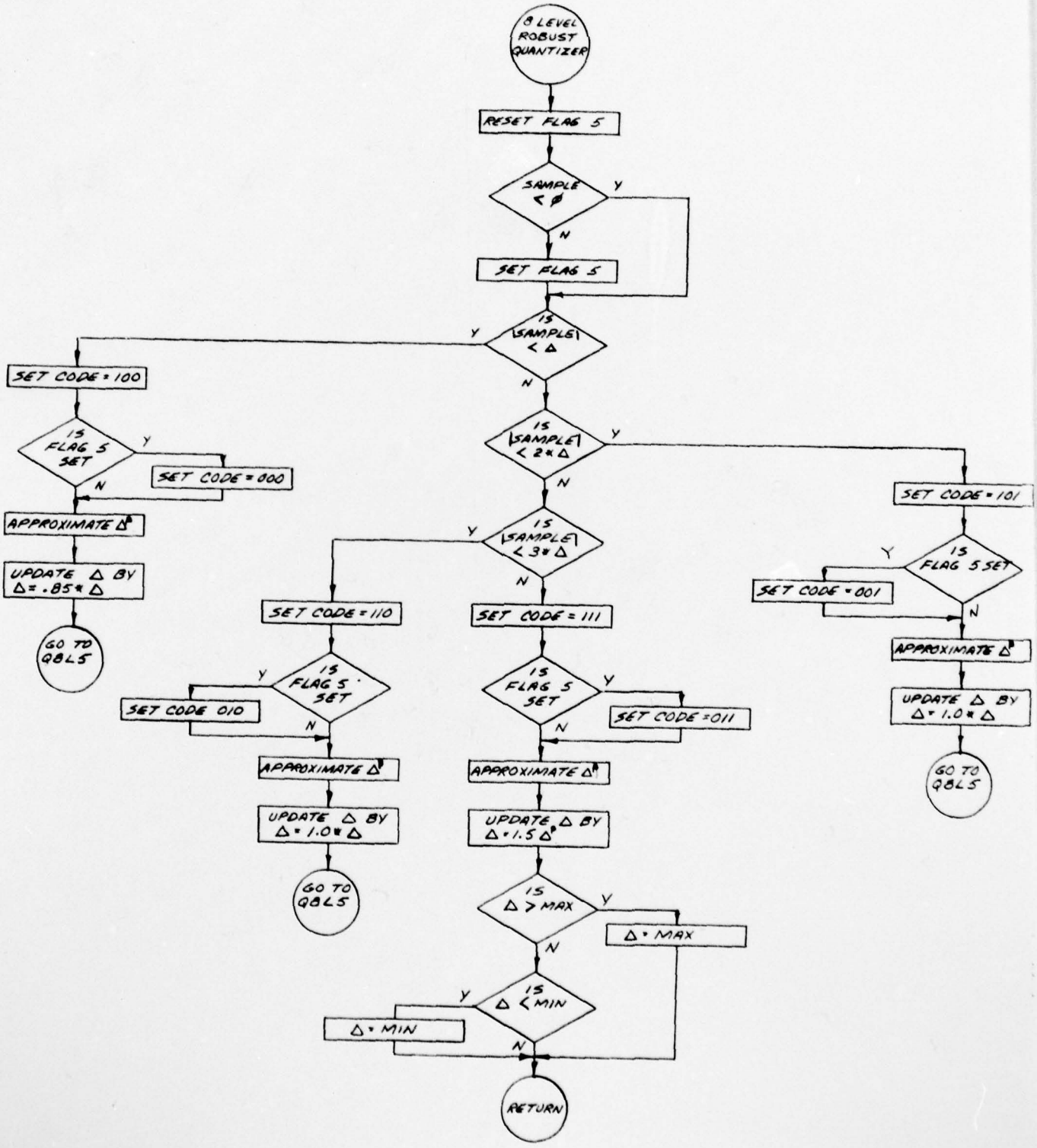


FIGURE 4-8 FLOW CHART OF AN 8-LEVEL ROBUST QUANTIZER

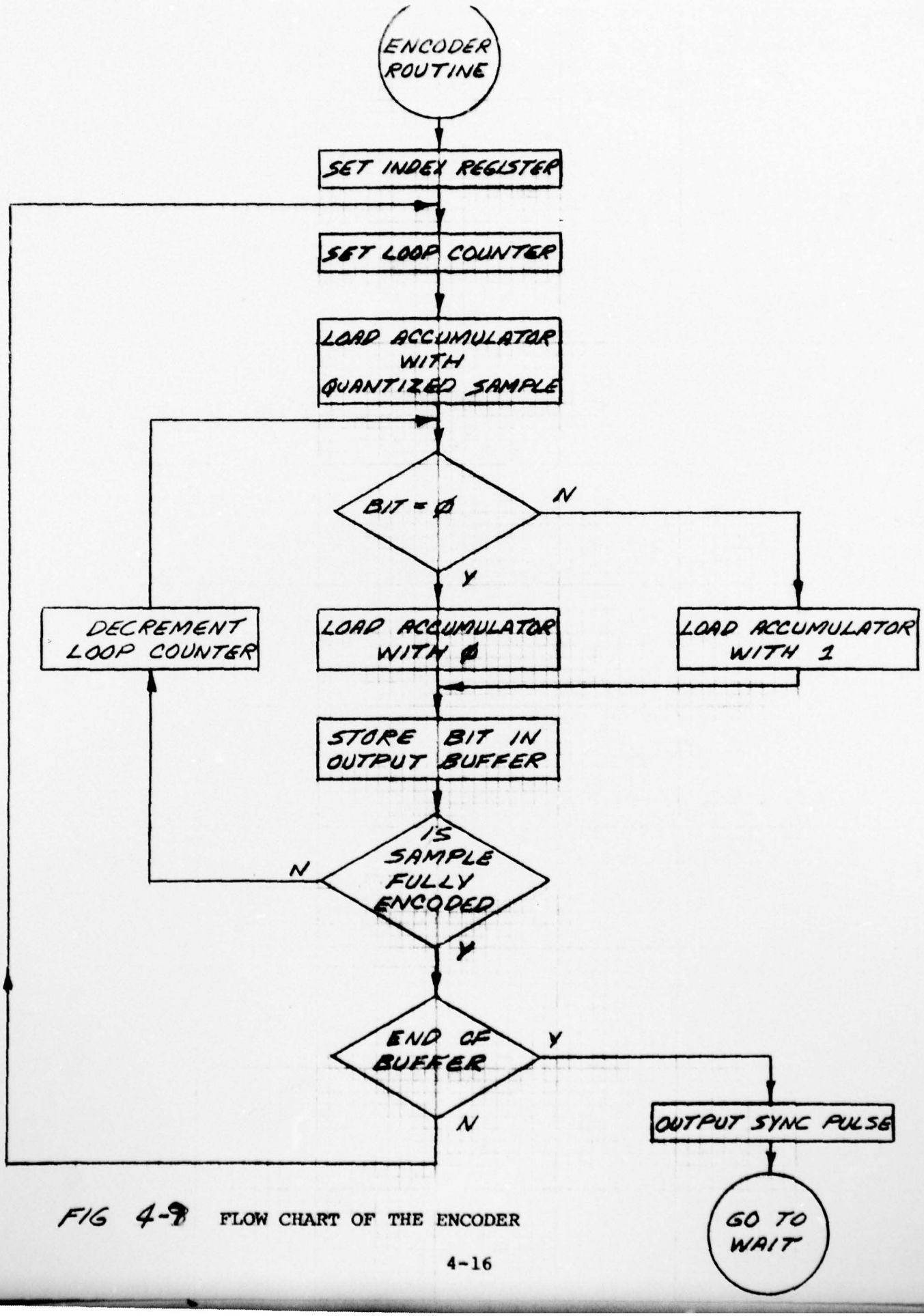


FIG 4-9 FLOW CHART OF THE ENCODER

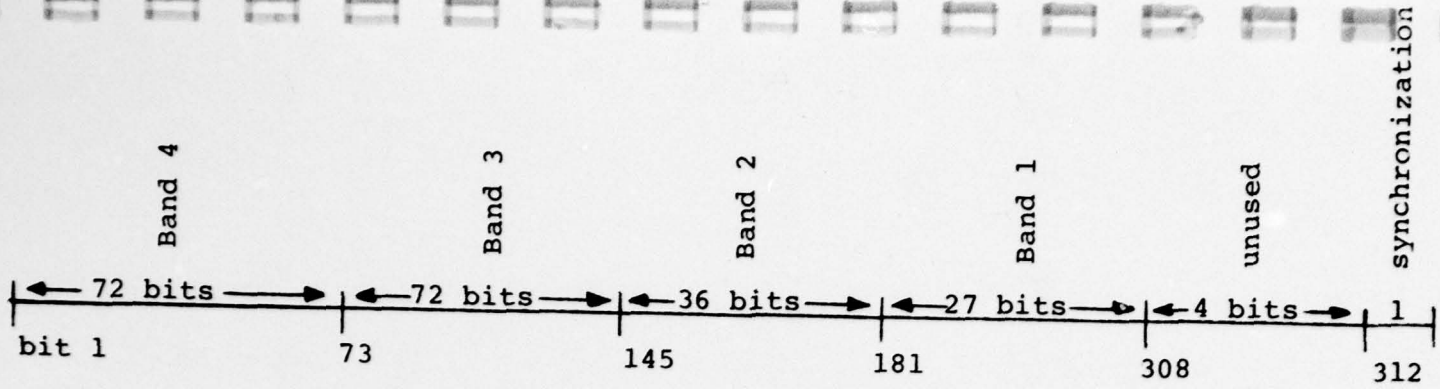


FIGURE 4-10(a) TRANSMISSION DATA FORMAT FOR 9.6 Kb/s

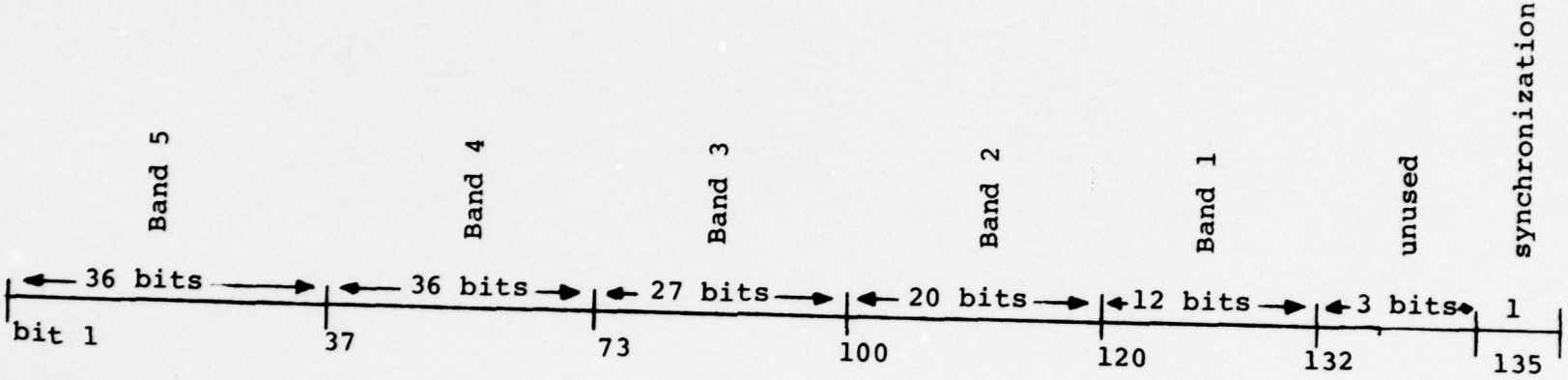


FIGURE 4-10(b) TRANSMISSION DATA FORMAT FOR 16.0 Kb/s

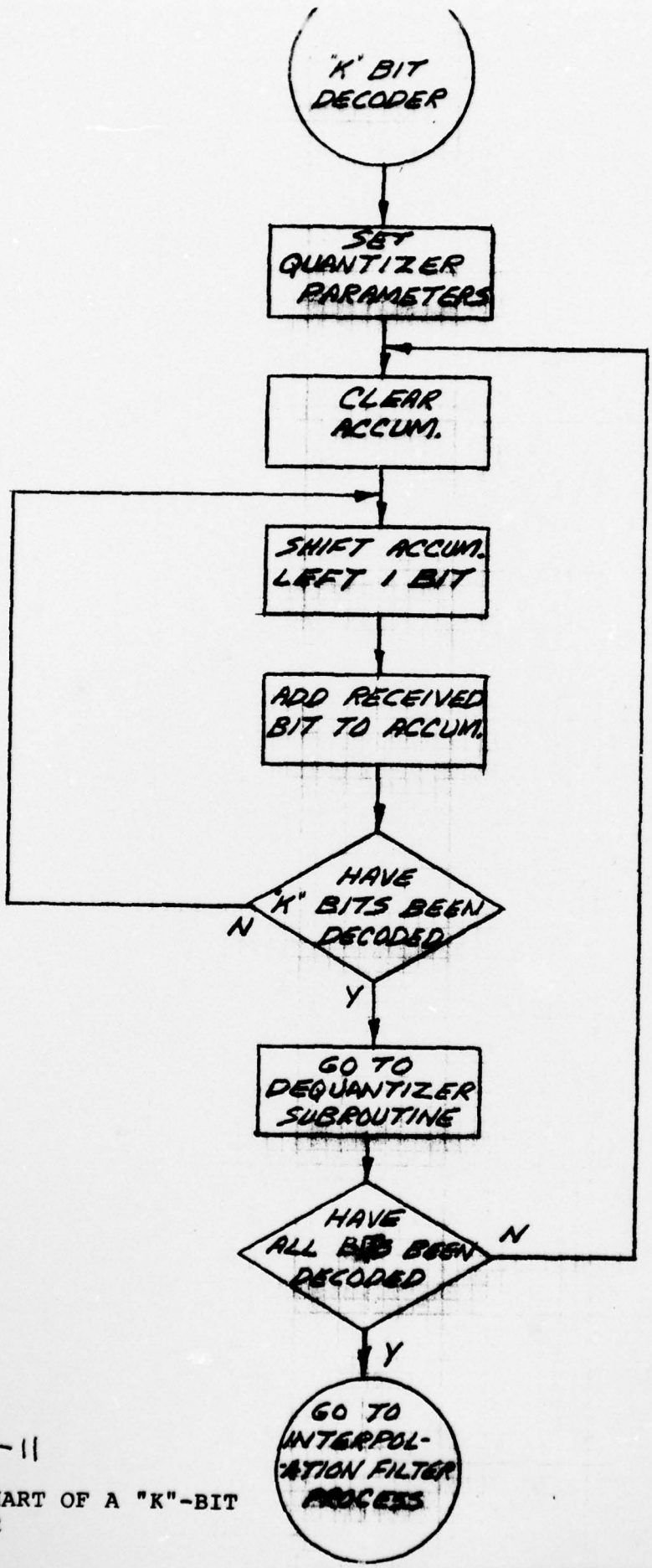


FIG 4-11

FLOW CHART OF A "K"-BIT DECODER

flow chart description of an 8-level robust dequantizer is depicted in Figure 4-13. Whenever the dequantizer step size Δ falls below a pre-determined threshold value Δth, a mid-rise/mid-tread switch, as described in Section 2, is used to eliminate the tones generated during idle channel conditions.

4.8 Interpolation Filtering

After dequantizing the received samples, they are stored in the approporate filter buffers. Since the received sample sequence contains information on the original signal only at intervals of L samples, where L is the decimation (interpolation) ratio, the remaining L-1 samples are generated by interpolation filtering at the receiver as shown in Figure 4-13.

The real-time interpolation filter code implements the following:

$$y(n) = \sum_{k=0}^{199} h(k)\, x\left[\frac{n-k}{L}\right] \tag{4-2}$$

where $[\xi]$ is defined as:

$$[\xi] = \begin{cases} \xi \ ; \text{ if } \xi \text{ is an integer} \\ 0 \ ; \text{ otherwise} \end{cases} \tag{4-3}$$

As discussed in Section 2, it is not necessary to perform 200 multiplies to obtain each output sample owing to the fact that a lot of the input points $x(\cdot)$ are zero. For the sake of clarity, the following interpolations of Band 1 signal in the 9.6 Kb/s system are examined in detail (L=20):

$$\begin{aligned}
&y(0) = h(0)\ x\ (0) + h(20)\ x\ (-1) + \ldots + h(180)\ x\ (-9) \\
&y(1) = h(1)\ x\ (0) + h(21)\ x\ (-1) + \ldots + h(181)\ x\ (-9) \\
&\vdots \\
&y(19) = h(19)\ x\ (0) + h(39)\ x\ (-1) + \ldots + h(199)\ x\ (-9) \\
&y(20) = h(0)\ x\ (1) + h(20)\ x\ (0) + \ldots + h(180)\ x\ (-8) \\
&y(21) = h(1)\ x\ (1) + h(21)\ x\ (0) + \ldots + h(181)\ x\ (-8) \\
&\vdots
\end{aligned} \tag{4-4}$$

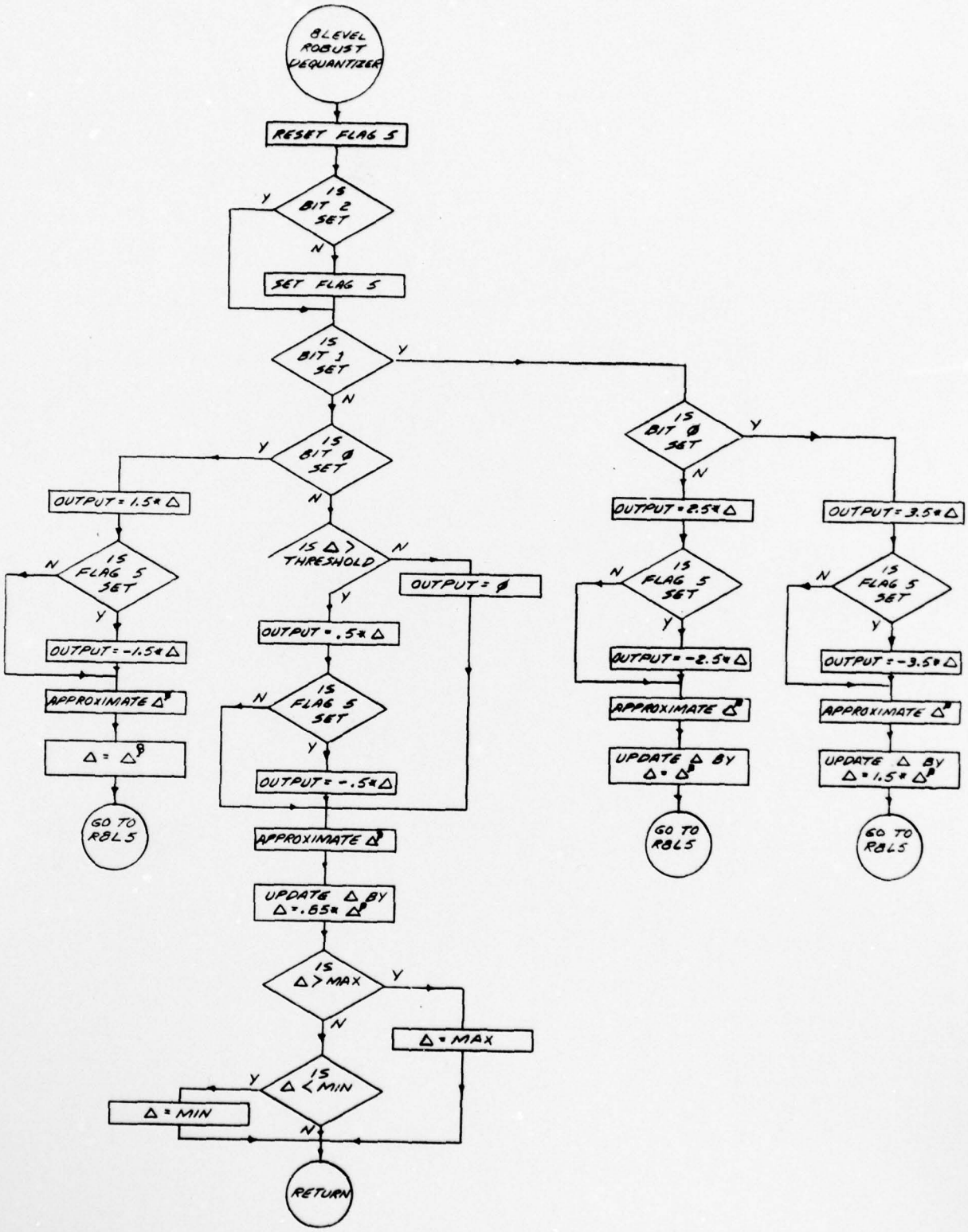


FIGURE 4-12 FLOW CHART FOR AN 8-LEVEL ROBUST DEQUANTIZER

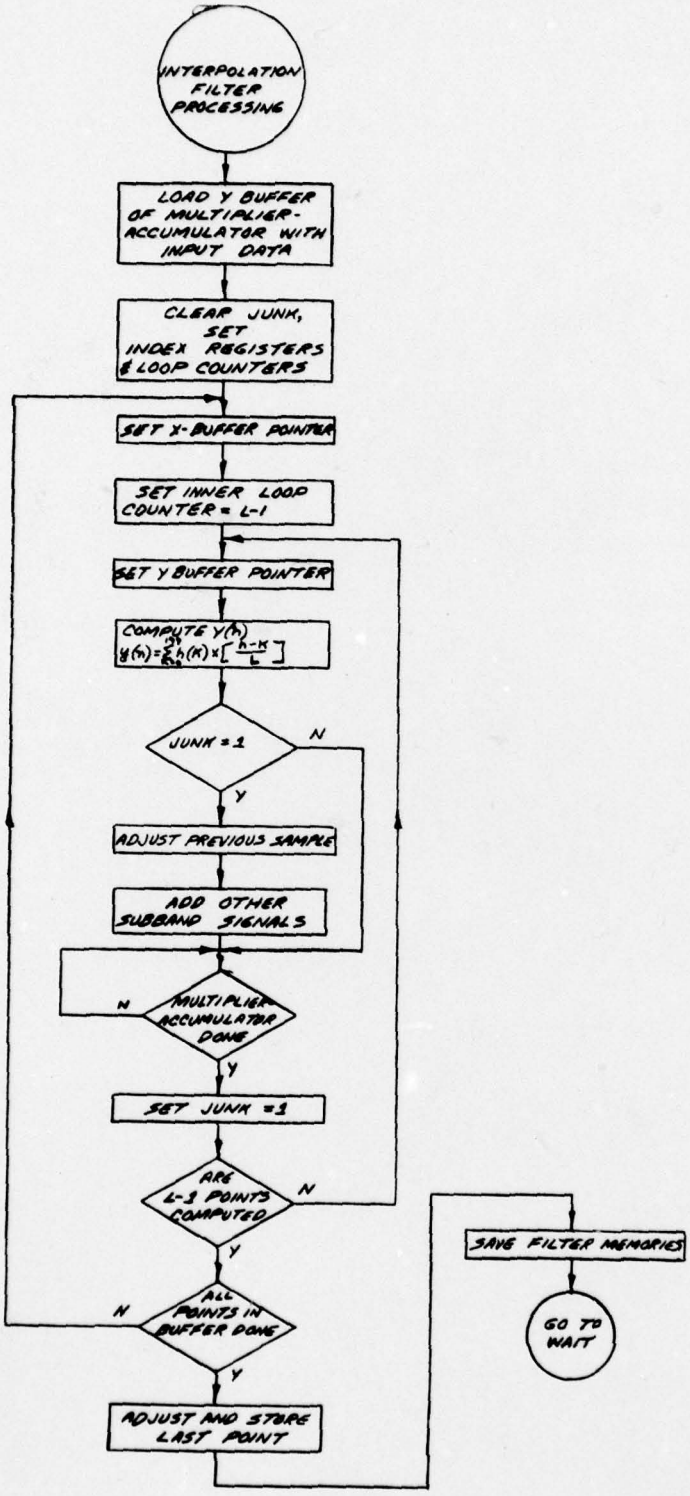


FIGURE 4-13 FLOW CHART FOR INTERPOLATION FILTERING

$$y(39) = h(19) \times (1) + h(39) \times (0) + \ldots + h(199) \times (-8)$$

$$\vdots$$

$$y(179) = h(19) \times (8) + h(39) \times (7) + \ldots + h(199) \times (-1)$$

It is obvious from the above equations that only 200/20 multiplications are needed to calculate each y(n). Furthermore, by arranging the filter coefficients and input samples in a special fashion, as shown in Figure 4-11, the output values can be efficiently computed using the MULACC.

Initially, the Y-buffer pointer is set to X(Ø) and the X-buffer pointer is set to h(Ø). The multiplier is set up to perform ten multiply-and-accumulates with both the X and Y pointers incrementing after each operation. When MULACC finishes, y(0) is computed. Then with the X-buffer pointer directing at h(1) and resetting the Y-buffer pointer to x(0), MULACC is restarted which later yields y(1).

The process is iterated for L-2 times and y(2), y(3) ... y(19) are computed. Then the X-buffer pointer is reset to h(0) and the Y-buffer pointer is set to x(1). By repeating the above procedure, all interpolator outputs are computed. Tallying up the multiplies involved, it is noted that the total number is roughly comparable in both interpolation and decimation filtering. However, a closer examination of the real-time implementation reveals that that symmetry of the FIR filters has not been exploited. This can be explained by the fact that programming of MULACC is more efficient when the filter coefficients are strung out. Furthermore, the speed of MULACC is capable of performing full-duplex 9.6 Kb/s subband coder and the more complicated 16.0 Kb/s system in a half-duplex mode.

When interpolation filterings for all subbands are finished, the resulting points are added together before being loaded into the output buffer. Then a sync pulse is generated to signify the end of the receiver function and the program returns to the wait loop.

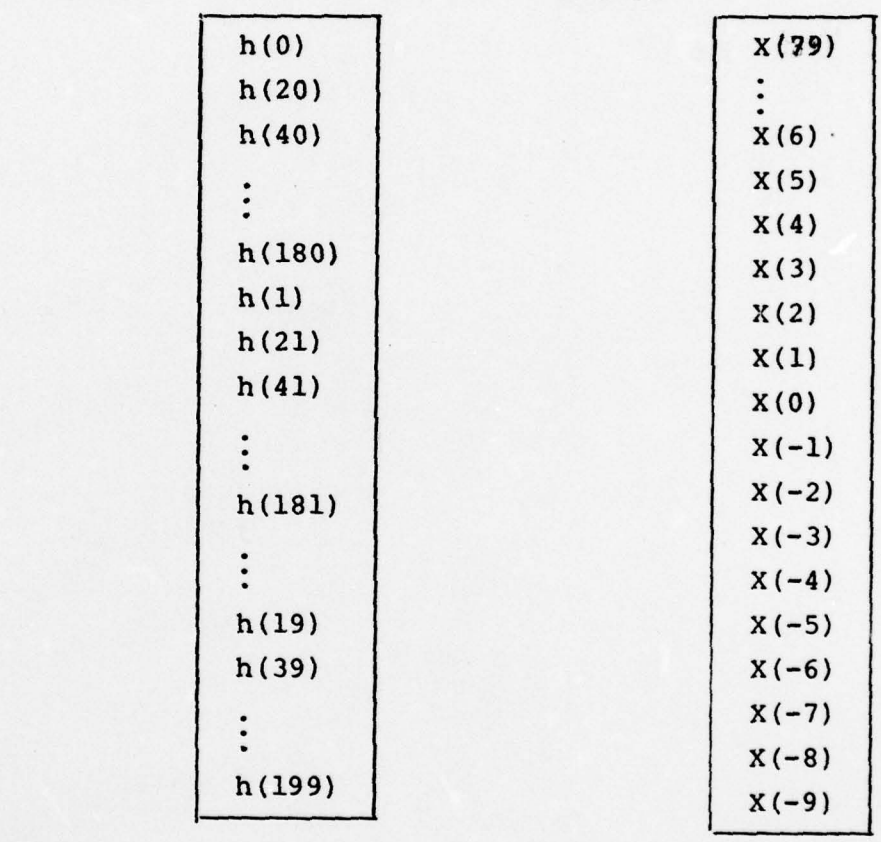


FIGURE 4-14 STORAGE OF X AND Y-BUFFERS FOR INTERPOLATION FILTERING

4.9 Operating Procedures of the Real-Time Subband Coder Programs

4.9.1 9.6 Kb/s Program

I. LOADING

Load the program SBC96.PBJ into both PSP's using the PSP Loader Program PSPLD.

II. OPERATING PROCEDURES

After the program has been loaded, depress bit 15 of PSP front panel switches. Then start the subband coder program by setting the RUN/STOP switch to RUN. In this mode, the program will loop around an initialization routine which clears out data buffers and resets loop registers. When bit 15 is placed in the up position, the software will first enable synchronization between the two PSP's and then proceed to process speech using the subband coder algorithm. If re-synchronization is desired, hold bit 15 of both PSP's in the down position and return it one at a time to the up position. Options of the 9.6 Kb/s subband coder program are summarized as:

bit 15	DOWN:	INITIALIZATION
	UP:	NORMAL OPERATION
bit 12	DOWN:	ROBUST QUANTIZER
	UP:	ADAPTIVE JAYANT QUANTIZER

4.9.2 16.0 Kb/s Program

I. LOADING

Load the program SBC16.PBJ into both PSP's using the PSP Loader Program PSPLD.

II. OPERATING PROCEDURES

After the program has been loaded, depress bit 15 of PSP front panel switches. Then start the subband coder program by setting the RUN/STOP switch to RUN. In this mode, the program will loop around an initialization routine which clears out data buffers and resets loop registers. When bit 15 is placed in the up position, the software will first enable synchronization between the two PSP's and then proceed to process speech using the subband coder algorithm. If re-synchronization is desired, hold bit 15 of both PSP's in the down position and return it to the up position. Since the 16 Kb/s subband coder program only works in the half-duplex mode, the push-to-talk switch on the handset has to be depressed to allow speech transmission. Options of the 16.0 Kb/s subband coder program are summarized as:

bit 15	DOWN:	INITIALIZATION
	UP:	NORMAL OPERATION
bit 14	DOWN:	TRANSMITTER
	UP:	PUSH-TO-TALK
bit 13	DOWN:	RECEIVER
	UP:	PUSH-TO-TALK
bit 12	DOWN:	ROBUST QUANTIZER
	UP:	ADAPTIVE JAYANT QUANTIZER

SECTION V
Conclusions and Recommendations

5.1 Conclusions

This contract has resulted in the development of a 9.6 and a 16.0 Kb/s real-time subband coder which employ the integer-band sampling technique. Informal listening tests indicate that the speech quality of the subband coders is slightly "hollow" and noisy especially for the 9.6 Kb/s system, but its overall quality is significantly better than that of CVSD at the same data rate. However, when compared to other high quality 16 Kb/s speech encoding schemes, such as the Adaptive Predictive Coder with Adaptive Quantization (APCQ), the subband coder is clearly inferior. GTE Sylvania attributes this to the fact that the four or five bandpass filters employed by the subband coder algorithm are not adequate to represent the entire speech spectrum. This introduces spectral "gaps" which manifest the "hollowness" in the processed speech. Moreover, the ratio of subband energies fluctuates largely from frame to frame and quantization of them using adaptive PCM (APCM) with fixed bit allocations results in noisier output speech.

In light of the inadequacy of the integer-band sampling approach, GTE Sylvania developed an improved subband coding scheme using quadrature mirror filters (QMF). An adaptive bit allocation scheme is also incorporated to quantize the subband signals. FORTRAN simulations show that no significant distortions are introduced in the bandsplitting/reconstruction process using QMF filters and this results in "smoother" quality processed speech. Furthermore, the adaptative bit allocation scheme enables a more efficient encoding of the subband waveforms which eliminates a lot of quantization noise in the outputs speech.

5.2 Recommendations

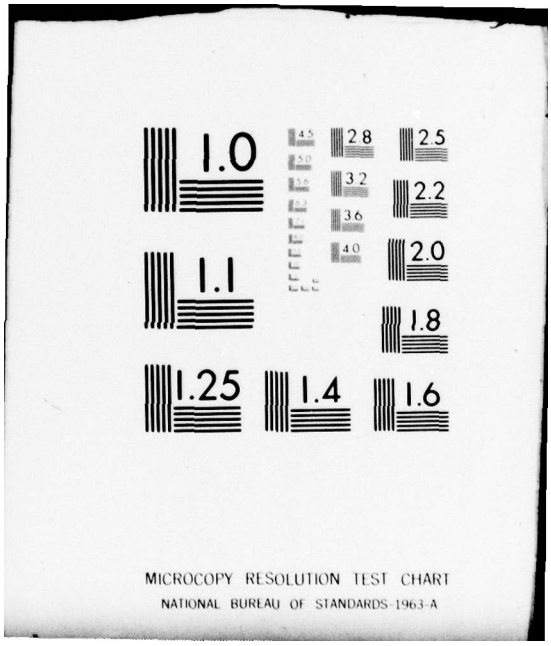
Encouraged by the latest results, GTE Sylvania strongly recommends the subband coding algorithm with the QMF approach be studied further and be implemented especially at 16 Kb/s. More-

over, since present subband coders are only designed to capture the spectral shape of the speech signal, GTE Sylvania believes that the speech quality of the QMF method can be further improved by exploiting the fine structures of the speech spectrum. This is also reinforced by the findings of a recent study which incorporates pitch information in performing subband coding of speech signals [14].

AD-A074 427 GTE SYLVANIA INC NEEDHAM HEIGHTS MASS ELECTRONIC SYS--ETC F/G 17/2
SUBBAND CODER STUDY.(U)
JUL 79 R S CHEUNG, R L WINSLOW DCA100-79-C-0001
UNCLASSIFIED SBIE-AD-E100 278 NL

2 OF 2
AD A074427

END
DATE FILMED
10-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

REFERENCES

[1] J. L. Flanagan et al, "Speech Coding," IEEE Trans Comm., Vol. COM-27, No. 4, pp 710-736, Apr., 1979.

[2] R. E. Crochiere, S. A. Webber and J. L. Flanagan, "Digital Coding of Speech in Subbands," Bell Syst. Tech. J., Vol. 55, No. 8, Oct., 1976.

[3] R. E. Crochiere, "On the Design of Subband Coders for Low Bit-Rate Speech Communication," Bell Syst. Tech. J., Vol. 56, No. 5, pp 747-770, May 1977.

[4] R. W. Shafer and L. R. Rabiner, "A Digital Signal Processing Approach to Interpolation," Proceedings of IEEE, Vol. 61, pp 692-702, June 1973.

[5] N. S. Jayant, "Digital Coding of Speech Waveform: PCM, DPCM and DM Quantizers," Proc. IEEE, Vol. 62, No. 5, May 1974.

[6] R. E. Crochiere, "A Mid-Rise/Mid-Tread Quantizer Switch for Improved Idle-Channel Performance in Adaptive Coders," Bell Syst. Tech. J., Vol. 56, No. 9, Oct. 1978.

[7] D. Esteban and C. Galand, "Applications of Quadrature Mirror Filters to Split Band Voice Coding Schemes," IEEE International Conference on Acoustics, Speech and Signal Processing, Hartford, Conn., 1977.

[8] O. Hermann, "On the Approximation Problem in Nonrecursive Digital Filter Design," IEEE Trans. Circuit Theory, Vol. CT-18, May 1971.

[9] L. E. Bergeron, "A Maximally Flat Filter Design Algorithm for Quadrature Mirror Filters," Conf. Rec. IEEE Int. Conf. Acoust., Speech and Sig. Proc., Washington, D.C., 1979.

[10] A. Crosier, D. Esteban and C. Galand, "Perfect Channel Splitting by Use of Interpolation/Decimation/Tree Decomposition Techniques," 1976 International Conference on Information Sciences and Systems, Patras.

[11] D. Esteban and C. Galand, "32 KBPS CCITT Compatible Split-Band Coding Scheme," Conf. Record of IEEE International Conference Acoustics, Speech and Signal Processing, Tulsa, OK, April 1978.

[12] J. Huang and P. Schultheiss, "Block Quantization of Correlated Gaussian Random Variables," IEEE Trans. Communications Systems, Vol. CS-11, 1963, pp 289-296.

[13] R. Zelinski and P. Noll, "Approaches to Adaptive Transform Speech Coding at Low Bit Rates," IEEE Trans. Acoust. Speech and Sig. Proc., Vol. ASSP-27, Feb. 1979.

[14] R. E. Crochiere, "A New Approach for Implementing Pitch Prediction in Subband Coding," Conf. Rec. IEEE International Conf. on Acoust., Speech and Sig. Proc., Washington, D.C., April 1979.

[15] D. J. Goodman and R. M. Wilkinson, "A Robust Quantizer," IEEE Trans. on Communications, Vol. COM-23, No. 11, Nov. 1975.

[16] H. G. Martinez and T. W. Parks, "A Class of Infinite-Duration Impulse Response Digital Filters for Sampling Rate Reduction," IEEE Trans. Acoust., Speech and Sig. Proc., Vol. ASSP-27, No. 2, April 1979.

A.1 FILTER COEFFICIENTS FOR BAND 1 OF THE 9.6 KB/S SUB-BAND CODER

FILENAME IS F1.NUM

H(1)= 0.24143001E-02=H(200)
H(2)=-0.11850999E-02=H(199)
H(3)=-0.77619997E-03=H(198)
H(4)=-0.39209999E-03=H(197)
H(5)=-0.46300000E-04=H(196)
H(6)= 0.24970001E-03=H(195)
H(7)= 0.48940000E-03=H(194)
H(8)= 0.66289998E-03=H(193)
H(9)= 0.76249999E-03=H(192)
H(10)= 0.77649998E-03=H(191)
H(11)= 0.69660001E-03=H(190)
H(12)= 0.51370001E-03=H(189)
H(13)= 0.22710000E-03=H(188)
H(14)=-0.16140001E-03=H(187)
H(15)=-0.63630001E-03=H(186)
H(16)=-0.11775000E-02=H(185)
H(17)=-0.17495001E-02=H(184)
H(18)=-0.23149000E-02=H(183)
H(19)=-0.28214001E-02=H(182)
H(20)=-0.32258001E-02=H(181)
H(21)=-0.34689000E-02=H(180)
H(22)=-0.35206000E-02=H(179)
H(23)=-0.33376999E-02=H(178)
H(24)=-0.29042000E-02=H(177)
H(25)=-0.22293001E-02=H(176)
H(26)=-0.13233999E-02=H(175)
H(27)=-0.22950000E-03=H(174)
H(28)= 0.99129998E-03=H(173)
H(29)= 0.22619001E-02=H(172)
H(30)= 0.35002001E-02=H(171)
H(31)= 0.46167001E-02=H(170)
H(32)= 0.55243000E-02=H(169)
H(33)= 0.61448999E-02=H(168)
H(34)= 0.64197001E-02=H(167)
H(35)= 0.63139000E-02=H(166)
H(36)= 0.58201998E-02=H(165)
H(37)= 0.49601998E-02=H(164)
H(38)= 0.37837999E-02=H(163)
H(39)= 0.23679000E-02=H(162)
H(40)= 0.80939999E-03=H(161)
H(41)=-0.77990000E-03=H(160)
H(42)=-0.22853001E-02=H(159)
H(43)=-0.35991000E-02=H(158)
H(44)=-0.46256999E-02=H(157)
H(45)=-0.53015999E-02=H(156)
H(46)=-0.55860002E-02=H(155)
H(47)=-0.54783998E-02=H(154)
H(48)=-0.50134002E-02=H(153)
H(49)=-0.42591998E-02=H(152)
H(50)=-0.33102999E-02=H(151)
H(51)=-0.22835000E-02=H(150)
H(52)=-0.13026000E-02=H(149)
H(53)=-0.48700001E-03=H(148)
H(54)= 0.61799998E-04=H(147)
H(55)= 0.26979999E-03=H(146)
H(56)= 0.10160000E-03=H(145)

A.1 (cont'd)

```
H( 57)=  0.43380001E-03*H(144)
H( 58)=  0.12800000E-02*H(143)
H( 59)=  0.23777000E-02*H(142)
H( 60)=  0.34956001E-02*H(141)
H( 61)=  0.45135595E-02*H(140)
H( 62)=  0.52355595E-02*H(139)
H( 63)=  0.56420001E-02*H(138)
H( 64)=  0.54104595E-02*H(137)
H( 65)=  0.44956595E-02*H(136)
H( 66)=  0.28458595E-02*H(135)
H( 67)=  0.47685595E-03*H(134)
H( 68)=  0.25237000E-02*H(133)
H( 69)=  0.59974595E-02*H(132)
H( 70)=  0.97220595E-02*H(131)
H( 71)=  0.13422200E-01*H(130)
H( 72)=  0.16791601E-01*H(129)
H( 73)=  0.19514700E-01*H(128)
H( 74)=  0.21293601E-01*H(127)
H( 75)=  0.21800595E-01*H(126)
H( 76)=  0.21053400E-01*H(125)
H( 77)=  0.18751901E-01*H(124)
H( 78)=  0.14563400E-01*H(123)
H( 79)=  0.90051000E-02*H(122)
H( 80)=  0.35055000E-02*H(121)
H( 81)=  0.30066000E-02*H(120)
H( 82)=  0.11116500E-01*H(119)
H( 83)=  0.18533000E-01*H(118)
H( 84)=  0.25417595E-01*H(117)
H( 85)=  0.31219000E-01*H(116)
H( 86)=  0.35512600E-01*H(115)
H( 87)=  0.37925000E-01*H(114)
H( 88)=  0.38212400E-01*H(113)
H( 89)=  0.36233000E-01*H(112)
H( 90)=  0.32023859E-01*H(111)
H( 91)=  0.25746901E-01*H(110)
H( 92)=  0.17727000E-01*H(109)
H( 93)=  0.83940001E-02*H(108)
H( 94)=  0.16958000E-02*H(107)
H( 95)=  0.11943200E-01*H(106)
H( 96)=  0.21716001E-01*H(105)
H( 97)=  0.30403501E-01*H(104)
H( 98)=  0.37458301E-01*H(103)
H( 99)=  0.42425501E-01*H(102)
H(100)=  0.44956195E-01*H(101)
```

A.2 FILTER COEFFICIENTS FOR BAND 2 OF THE 9.6 KB/S SUB-BAND CODER

FILENAME IS F2.NUM

H(1)=-0.28814999E-02=H(200)
H(2)=-0.48419999E-03=H(199)
H(3)=-0.17190000E-03=H(198)
H(4)= 0.30180000E-03=H(197)
H(5)= 0.78910001E-03=H(196)
H(6)= 0.11328000E-02=H(195)
H(7)= 0.12211000E-02=H(194)
H(8)= 0.10447000E-02=H(193)
H(9)= 0.70029998E-03=H(192)
H(10)= 0.36320000E-03=H(191)
H(11)= 0.20430000E-03=H(190)
H(12)= 0.32250001E-03=H(189)
H(13)= 0.67719998E-03=H(188)
H(14)= 0.10980000E-02=H(187)
H(15)= 0.13005000E-02=H(186)
H(16)= 0.12387000E-02=H(185)
H(17)= 0.88680000E-03=H(184)
H(18)=-0.11686001E-02=H(183)
H(19)=-0.27037000E-02=H(182)
H(20)=-0.39443001E-02=H(181)
H(21)=-0.44006002E-02=H(180)
H(22)=-0.37459999E-02=H(179)
H(23)=-0.19995000E-02=H(178)
H(24)= 0.60370000E-03=H(177)
H(25)= 0.33477000E-02=H(176)
H(26)= 0.55231000E-02=H(175)
H(27)= 0.65023000E-02=H(174)
H(28)= 0.59827999E-02=H(173)
H(29)= 0.40560002E-02=H(172)
H(30)= 0.13738000E-02=H(171)
H(31)=-0.14005000E-02=H(170)
H(32)=-0.34665000E-02=H(169)
H(33)=-0.43250001E-02=H(168)
H(34)=-0.39116000E-02=H(167)
H(35)=-0.26010999E-02=H(166)
H(36)=-0.10628999E-02=H(165)
H(37)= 0.21999999E-05=H(164)
H(38)= 0.13520000E-03=H(163)
H(39)=-0.68940001E-03=H(162)
H(40)=-0.20399000E-02=H(161)
H(41)=-0.31822999E-02=H(160)
H(42)=-0.33676000E-02=H(159)
H(43)=-0.21398000E-02=H(158)
H(44)= 0.43789999E-03=H(157)
H(45)= 0.37541001E-02=H(156)
H(46)= 0.68172999E-02=H(155)
H(47)= 0.85922005E-02=H(154)
H(48)= 0.83713001E-02=H(153)
H(49)= 0.60546999E-02=H(152)
H(50)= 0.22213999E-02=H(151)
H(51)=-0.20415999E-02=H(150)
H(52)=-0.55050999E-02=H(149)
H(53)=-0.72280001E-02=H(148)
H(54)=-0.68941000E-02=H(147)
H(55)=-0.49244999E-02=H(146)
H(56)=-0.23167999E-02=H(145)

A.2 (cont'd)

H(57)=-0.25210000E-03-H(144)
H(58)= 0.37679999E-03-H(143)
H(59)=-0.65270002E-03-H(142)
H(60)=-0.27643000E-02-H(141)
H(61)=-0.47696000E-02-H(140)
H(62)=-0.53202999E-02-H(139)
H(63)=-0.34683000E-02-H(138)
H(64)= 0.87934000E-03-H(137)
H(65)= 0.67961002E-02-H(136)
H(66)= 0.12558000E-01-H(135)
H(67)= 0.16176401E-01-H(134)
H(68)= 0.16128501E-01-H(133)
H(69)= 0.11900600E-01-H(132)
H(70)= 0.44201993E-02-H(131)
H(71)= 0.44368000E-02-H(130)
H(72)=-0.12152500E-01-H(129)
H(73)=-0.16685001E-01-H(128)
H(74)=-0.15593704E-01-H(127)
H(75)=-0.12856200E-01-H(126)
H(76)=-0.65404003E-02-H(125)
H(77)=-0.75980002E-03-H(124)
H(78)= 0.40659976E-02-H(123)
H(79)= 0.10424995E-01-H(122)
H(80)= 0.15477997E-01-H(121)
H(81)= 0.15111601E-01-H(120)
H(82)= 0.10271298E-01-H(119)
H(83)= 0.28891998E-02-H(118)
H(84)= 0.18065000E-02-H(117)
H(85)= 0.10537700E-01-H(116)
H(86)= 0.19136601E-01-H(115)
H(87)= 0.21464500E-01-H(114)
H(88)= 0.44196900E-01-H(113)
H(89)= 0.35075001E-01-H(112)
H(90)= 0.14019500E-01-H(111)
H(91)=-0.15326900E-01-H(110)
H(92)=-0.46201300E-01-H(109)
H(93)=-0.70392199E-01-H(108)
H(94)=-0.80527298E-01-H(107)
H(95)=-0.72332203E-01-H(106)
H(96)=-0.46141900E-01-H(105)
H(97)=-0.71588000E-02-H(104)
H(98)= 0.38693701E-01-H(103)
H(99)= 0.72043501E-01-H(102)
H(100)= 0.92843197E-01-H(101)

A.3 FILTER COEFFICIENTS FOR BAND 3 OF THE 9.6 KB/S SUB-BAND CODER

FILENAME IS F3.NUM

H(1)=-0.62770001E-03=H(200)
H(2)= 0.10161001E-02=H(199)
H(3)=-0.16489000E-02=H(198)
H(4)=-0.10597000E-02=H(197)
H(5)= 0.82850002E-03=H(196)
H(6)= 0.26167000E-02=H(195)
H(7)= 0.30443999E-02=H(194)
H(8)= 0.14219000E-02=H(193)
H(9)=-0.15072000E-02=H(192)
H(10)=-0.36853999E-02=H(191)
H(11)=-0.32213000E-02=H(190)
H(12)=-0.93199997E-04=H(189)
H(13)= 0.35224998E-02=H(188)
H(14)= 0.48226002E-02=H(187)
H(15)= 0.26632000E-02=H(186)
H(16)=-0.13586000E-02=H(185)
H(17)=-0.48307000E-02=H(184)
H(18)=-0.56171001E-02=H(183)
H(19)=-0.32870003E-02=H(182)
H(20)= 0.21866998E-02=H(181)
H(21)= 0.81097999E-02=H(180)
H(22)= 0.17137000E-02=H(179)
H(23)=-0.18490000E-02=H(178)
H(24)=-0.00000000E-03=H(177)
H(25)=-0.12670000E-03=H(176)
H(26)= 0.38670000E-03=H(175)
H(27)=-0.69710001E-03=H(174)
H(28)=-0.24130000E-02=H(173)
H(29)=-0.27240999E-02=H(172)
H(30)=-0.34950001E-03=H(171)
H(31)= 0.34123999E-02=H(170)
H(32)= 0.54050002E-02=H(169)
H(33)= 0.34081000E-02=H(168)
H(34)=-0.15641000E-02=H(167)
H(35)=-0.57593998E-02=H(166)
H(36)=-0.58081001E-02=H(165)
H(37)=-0.16876000E-02=H(164)
H(38)= 0.32305998E-02=H(163)
H(39)= 0.51683998E-02=H(162)
H(40)= 0.32046000E-02=H(161)
H(41)=-0.21660000E-03=H(160)
H(42)=-0.19279000E-02=H(159)
H(43)=-0.11402000E-02=H(158)
H(44)= 0.72199997E-04=H(157)
H(45)=-0.61170000E-03=H(156)
H(46)=-0.28825001E-02=H(155)
H(47)=-0.37535999E-02=H(154)
H(48)=-0.75980002E-03=H(153)
H(49)= 0.48411000E-02=H(152)
H(50)= 0.83614001E-02=H(151)
H(51)= 0.57621002E-02=H(150)
H(52)=-0.21144000E-02=H(149)
H(53)=-0.94780996E-02=H(148)
H(54)=-0.10255500E-01=H(147)
H(55)=-0.34884000E-02=H(146)
H(56)= 0.55614999E-02=H(145)

A.3 (cont'd)

H(57)= 0.988369995E-02-H(144)
H(58)= 0.677399995E-02-H(143)
H(59)=-0.128000000E-03-H(142)
H(60)=-0.465210015E-02-H(141)
H(61)=-0.399369938E-02-H(140)
H(62)=-0.921100035E-03-H(139)
H(63)=-0.180799995E-03-H(138)
H(64)=-0.289019995E-02-H(137)
H(65)=-0.494210005E-02-H(136)
H(66)=-0.146039995E-02-H(135)
H(67)= 0.704230005E-02-H(134)
H(68)= 0.134911005E-01-H(133)
H(69)= 0.160439005E-01-H(132)
H(70)=-0.250040005E-02-H(131)
H(71)=-0.167021015E-01-H(130)
H(72)=-0.195104005E-01-H(129)
H(73)=-0.766769935E-02-H(128)
H(74)= 0.103545005E-01-H(127)
H(75)= 0.206899005E-01-H(126)
H(76)= 0.155259005E-01-H(125)
H(77)= 0.505099935E-03-H(124)
H(78)=-0.115044005E-01-H(123)
H(79)=-0.124125005E-01-H(122)
H(80)=-0.431430015E-02-H(121)
H(81)= 0.105030005E-02-H(120)
H(82)=-0.176370005E-02-H(119)
H(83)=-0.699865015E-02-H(118)
H(84)=-0.312940015E-02-H(117)
H(85)= 0.128431005E-01-H(116)
H(86)= 0.284658005E-01-H(115)
H(87)= 0.246002955E-01-H(114)
H(88)=-0.517289995E-02-H(113)
H(89)=-0.429479005E-01-H(112)
H(90)=-0.585944025E-01-H(111)
H(91)=-0.256328015E-01-H(110)
H(92)= 0.324894005E-01-H(109)
H(93)= 0.777852975E-01-H(108)
H(94)= 0.698063005E-01-H(107)
H(95)= 0.672549995E-02-H(106)
H(96)=-0.688515015E-01-H(105)
H(97)=-0.996595995E-01-H(104)
H(98)=-0.585790025E-01-H(103)
H(99)= 0.275668005E-01-H(102)
H(100)= 0.961591975E-01-H(101)

A.4 FILTER COEFFICIENTS FOR BAND 4 OF THE 9.6 KB/S SUB-BAND CODER

FILENAME IS F4.NUM

H(1)=-0.59439999E-03=H(200)
H(2)= 0.17949000E-02=H(199)
H(3)= 0.20870000E-03=H(198)
H(4)= 0.13007000E-02=H(197)
H(5)= 0.10491000E-02=H(196)
H(6)=-0.22382000E-02=H(195)
H(7)=-0.18011000E-02=H(194)
H(8)= 0.37324000E-02=H(193)
H(9)= 0.26950999E-02=H(192)
H(10)=-0.50025000E-02=H(191)
H(11)=-0.33245999E-02=H(190)
H(12)= 0.55694999E-02=H(189)
H(13)= 0.34041000E-02=H(188)
H(14)=-0.58059001E-02=H(187)
H(15)=-0.27762000E-02=H(186)
H(16)= 0.34857001E-02=H(185)
H(17)= 0.15006000E-02=H(184)
H(18)=-0.11021000E-02=H(183)
H(19)= 0.12919999E-03=H(182)
H(20)=-0.14381000E-02=H(181)
H(21)=-0.16450000E-02=H(180)
H(22)= 0.33640000E-02=H(179)
H(23)= 0.25527001E-02=H(178)
H(24)=-0.48250998E-02=H(177)
H(25)=-0.35247999E-02=H(176)
H(26)= 0.31546999E-02=H(175)
H(27)= 0.14833000E-02=H(174)
H(28)=-0.97870000E-03=H(173)
H(29)= 0.26000000E-03=H(172)
H(30)=-0.17773000E-02=H(171)
H(31)=-0.21085001E-02=H(170)
H(32)= 0.41140998E-02=H(169)
H(33)= 0.30999001E-02=H(168)
H(34)=-0.50999001E-02=H(167)
H(35)=-0.34525001E-02=H(166)
H(36)= 0.42178002E-02=H(165)
H(37)= 0.22064999E-02=H(164)
H(38)=-0.16150000E-02=H(163)
H(39)= 0.68499998E-04=H(162)
H(40)=-0.18921000E-02=H(161)
H(41)=-0.26251001E-02=H(160)
H(42)= 0.50598001E-02=H(159)
H(43)= 0.44998000E-02=H(158)
H(44)=-0.66292998E-02=H(157)
H(45)=-0.48727999E-02=H(156)
H(46)= 0.58221999E-02=H(155)
H(47)= 0.33906000E-02=H(154)
H(48)=-0.26759999E-02=H(153)
H(49)=-0.37450000E-03=H(152)
H(50)=-0.18675000E-02=H(151)
H(51)=-0.32144999E-02=H(150)
H(52)= 0.62250998E-02=H(149)
H(53)= 0.60557998E-02=H(148)
H(54)=-0.86965999E-02=H(147)
H(55)=-0.69327001E-02=H(146)
H(56)= 0.81043001E-02=H(145)

A.4 (cont'd)

H(57)= 0.52148001E-02=H(144)
H(58)=-0.43012998E-02=H(143)
H(59)=-0.11896000E-02=H(142)
H(60)=-0.16517000E-02=H(141)
H(61)=-0.39189002E-02=H(140)
H(62)= 0.77372999E-02=H(139)
H(63)= 0.82810000E-02=H(138)
H(64)=-0.11624100E-01=H(137)
H(65)=-0.10068500E-01=H(136)
H(66)= 0.11511300E-01=H(135)
H(67)= 0.61588998E-02=H(134)
H(68)=-0.68791001E-02=H(133)
H(69)=-0.26785999E-02=H(132)
H(70)=-0.11483000E-02=H(131)
H(71)=-0.48461002E-02=H(130)
H(72)= 0.95886004E-02=H(129)
H(73)= 0.11834400E-01=H(128)
H(74)=-0.16348500E-01=H(127)
H(75)=-0.15428400E-01=H(126)
H(76)= 0.17335000E-01=H(125)
H(77)= 0.13523000E-01=H(124)
H(78)=-0.11885200E-01=H(123)
H(79)=-0.66884002E-02=H(122)
H(80)=-0.80000000E-05=H(121)
H(81)=-0.63705998E-02=H(120)
H(82)= 0.14192900E-01=H(119)
H(83)= 0.18966500E-01=H(118)
H(84)=-0.26076701E-01=H(117)
H(85)=-0.27186399E-01=H(116)
H(86)= 0.30435801E-01=H(115)
H(87)= 0.26515000E-01=H(114)
H(88)=-0.23065999E-01=H(113)
H(89)=-0.14009600E-01=H(112)
H(90)= 0.33447000E-02=H(111)
H(91)=-0.10430300E-01=H(110)
H(92)= 0.27459901E-01=H(109)
H(93)= 0.43680899E-01=H(108)
H(94)=-0.63997798E-01=H(107)
H(95)=-0.79856999E-01=H(106)
H(96)= 0.99164702E-01=H(105)
H(97)= 0.11182480E+00=H(104)
H(98)=-0.12560840E+00=H(103)
H(99)=-0.13261910E+00=H(102)
H(100)= 0.13761920E+00=H(101)

APPENDIX B

B.1 FILTER COEFFICIENTS FOR BAND 1 OF THE 16. KB/S SUB-BAND CODER

FILENAME IS FILT1.NUM

H(1)= 0.33366000E-02=H(200)
H(2)= 0.27339999E-03=H(199)
H(3)= 0.26400001E-03=H(198)
H(4)= 0.24300000E-03=H(197)
H(5)= 0.20890000E-03=H(196)
H(6)= 0.16420000E-03=H(195)
H(7)= 0.11170000E-03=H(194)
H(8)= 0.54700002E-04=H(193)
H(9)=-0.27999998E-05=H(192)
H(10)=-0.56400000E-04=H(191)
H(11)=-0.10130000E-03=H(190)
H(12)=-0.13299999E-03=H(189)
H(13)=-0.14760000E-03=H(188)
H(14)=-0.14280000E-03=H(187)
H(15)=-0.11850000E-03=H(186)
H(16)=-0.77999997E-04=H(185)
H(17)=-0.49999998E-05=H(184)
H(18)= 0.79600002E-04=H(183)
H(19)= 0.18000001E-03=H(182)
H(20)= 0.29020000E-03=H(181)
H(21)= 0.40230001E-03=H(180)
H(22)= 0.50899997E-03=H(179)
H(23)= 0.59960002E-03=H(178)
H(24)= 0.66580001E-03=H(177)
H(25)= 0.69740001E-03=H(176)
H(26)= 0.68469997E-03=H(175)
H(27)= 0.61849999E-03=H(174)
H(28)= 0.49150002E-03=H(173)
H(29)= 0.32040000E-03=H(172)
H(30)= 0.36900001E-04=H(171)
H(31)=-0.29249999E-03=H(170)
H(32)=-0.69100002E-03=H(169)
H(33)=-0.11469000E-02=H(168)
H(34)=-0.16531000E-02=H(167)
H(35)=-0.21962000E-02=H(166)
H(36)=-0.27562999E-02=H(165)
H(37)=-0.33163000E-02=H(164)
H(38)=-0.38626999E-02=H(163)
H(39)=-0.43418999E-02=H(162)
H(40)=-0.47581000E-02=H(161)
H(41)=-0.50760000E-02=H(160)
H(42)=-0.52709999E-02=H(159)
H(43)=-0.53204000E-02=H(158)
H(44)=-0.52048001E-02=H(157)
H(45)=-0.49092001E-02=H(156)
H(46)=-0.44228998E-02=H(155)
H(47)=-0.37400001E-02=H(154)
H(48)=-0.28643999E-02=H(153)
H(49)=-0.18036000E-02=H(152)
H(50)=-0.57430001E-03=H(151)
H(51)= 0.79999998E-03=H(150)
H(52)= 0.22887001E-02=H(149)
H(53)= 0.38548999E-02=H(148)
H(54)= 0.54559000E-02=H(147)
H(55)= 0.70443000E-02=H(146)
H(56)= 0.85696001E-02=H(145)

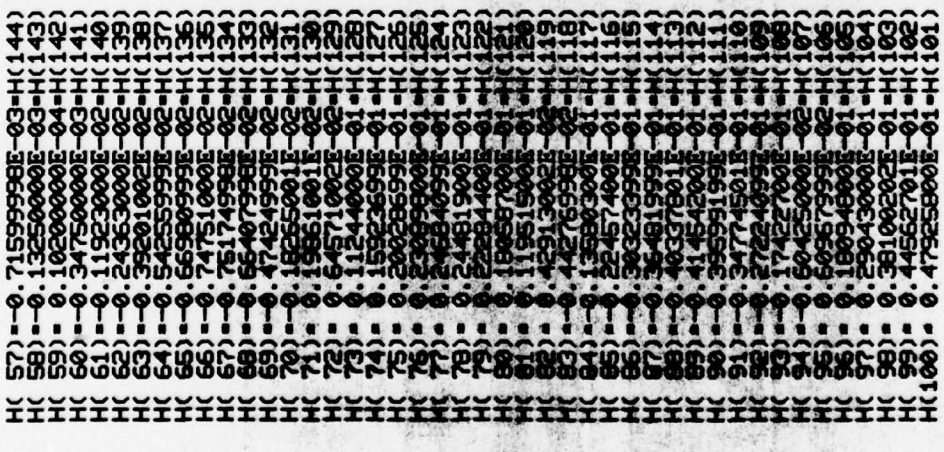
B.1 (cont'd)

B.2 FILTER COEFFICIENTS FOR BAND 2 OF THE 16. KB/S SUB-BAND CODER

FILENAME IS FILT2.NUM

H(1)=-0.165520000E-02=H(200)
H(2)= 0.189330000E-02=H(199)
H(3)= 0.115090000E-02=H(198)
H(4)= 0.723599998E-03=H(197)
H(5)= 0.463500001E-03=H(196)
H(6)= 0.289599999E-03=H(195)
H(7)= 0.163499999E-03=H(194)
H(8)= 0.710000002E-04=H(193)
H(9)= 0.139000000E-04=H(192)
H(10)= 0.200000000E-06=H(191)
H(11)= 0.401000000E-04=H(190)
H(12)= 0.139500000E-03=H(189)
H(13)= 0.298700000E-03=H(188)
H(14)= 0.507000002E-03=H(187)
H(15)= 0.744900003E-03=H(186)
H(16)= 0.981700002E-03=H(185)
H(17)= 0.118000000E-02=H(184)
H(18)= 0.130000000E-02=H(183)
H(19)= 0.130070000E-02=H(182)
H(20)= 0.114830000E-02=H(181)
H(21)= 0.823099998E-03=H(180)
H(22)= 0.319299999E-03=H(179)
H(23)=-0.345100001E-03=H(178)
H(24)=-0.113610000E-02=H(177)
H(25)=-0.198989999E-02=H(176)
H(26)=-0.284120000E-02=H(175)
H(27)=-0.359319999E-02=H(174)
H(28)=-0.416810002E-02=H(173)
H(29)=-0.448089998E-02=H(172)
H(30)=-0.446090001E-02=H(171)
H(31)=-0.407620001E-02=H(170)
H(32)=-0.331430000E-02=H(169)
H(33)=-0.219940000E-02=H(168)
H(34)=-0.795500000E-03=H(167)
H(35)= 0.798000002E-03=H(166)
H(36)= 0.246110000E-02=H(165)
H(37)= 0.405710000E-02=H(164)
H(38)= 0.544629999E-02=H(163)
H(39)= 0.649680001E-02=H(162)
H(40)= 0.710430000E-02=H(161)
H(41)= 0.720140001E-02=H(160)
H(42)= 0.676750002E-02=H(159)
H(43)= 0.582889998E-02=H(158)
H(44)= 0.446170001E-02=H(157)
H(45)= 0.278300000E-02=H(156)
H(46)= 0.944000003E-03=H(155)
H(47)=-0.888900002E-03=H(154)
H(48)=-0.254830000E-02=H(153)
H(49)=-0.388880000E-02=H(152)
H(50)=-0.479750001E-02=H(151)
H(51)=-0.521449998E-02=H(150)
H(52)=-0.513300002E-02=H(149)
H(53)=-0.460659999E-02=H(148)
H(54)=-0.374320001E-02=H(147)
H(55)=-0.268930001E-02=H(146)
H(56)=-0.162120000E-02=H(145)

B.2 (cont'd)



B.3 FILTER COEFFICIENTS FOR BAND 3 OF THE 16. KB/S SUB-BAND CODER

FILENAME IS FILT3.NUM

H(1)=-0.235960000E-02=H(200)
H(2)=-0.168700000E-03=H(199)
H(3)= 0.112000000E-04=H(198)
H(4)= 0.237900001E-03=H(197)
H(5)= 0.412600000E-03=H(196)
H(6)= 0.457999999E-03=H(195)
H(7)= 0.356800001E-03=H(194)
H(8)= 0.172400000E-03=H(193)
H(9)= 0.287999999E-04=H(192)
H(10)= 0.651000001E-04=H(191)
H(11)= 0.366700000E-03=H(190)
H(12)= 0.913999997E-03=H(189)
H(13)= 0.155590000E-02=H(188)
H(14)= 0.204300000E-02=H(187)
H(15)= 0.209459999E-02=H(186)
H(16)= 0.151740000E-02=H(185)
H(17)= 0.305200000E-03=H(184)
H(18)=-0.134500000E-02=H(183)
H(19)=-0.302190001E-02=H(182)
H(20)=-0.423510000E-02=H(181)
H(21)=-0.455040021E-02=H(180)
H(22)=-0.375130000E-02=H(179)
H(23)=-0.192870000E-02=H(178)
H(24)= 0.510399999E-03=H(177)
H(25)= 0.296010000E-02=H(176)
H(26)= 0.474169999E-02=H(175)
H(27)= 0.541129999E-02=H(174)
H(28)= 0.481210000E-02=H(173)
H(29)= 0.317990001E-02=H(172)
H(30)= 0.105120000E-02=H(171)
H(31)=-0.910000000E-03=H(170)
H(32)=-0.214560000E-02=H(169)
H(33)=-0.239780011E-02=H(168)
H(34)=-0.179000000E-02=H(167)
H(35)=-0.780000000E-03=H(166)
H(36)= 0.272000000E-04=H(165)
H(37)= 0.120500000E-03=H(164)
H(38)=-0.695800000E-03=H(163)
H(39)=-0.219800000E-02=H(162)
H(40)=-0.379609999E-02=H(161)
H(41)=-0.473870011E-02=H(160)
H(42)=-0.439780023E-02=H(159)
H(43)=-0.253430011E-02=H(158)
H(44)= 0.563099998E-03=H(157)
H(45)= 0.412160022E-02=H(156)
H(46)= 0.711739999E-02=H(155)
H(47)= 0.861180011E-02=H(154)
H(48)= 0.808439996E-02=H(153)
H(49)= 0.563610022E-02=H(152)
H(50)= 0.197630000E-02=H(151)
H(51)=-0.180380000E-02=H(150)
H(52)=-0.460750000E-02=H(149)
H(53)=-0.571860000E-02=H(148)
H(54)=-0.505860011E-02=H(147)
H(55)=-0.321089999E-02=H(146)
H(56)=-0.119310000E-02=H(145)

B.3 (cont'd)

H(57)=-0.521000000E-04=H(144)
H(58)=-0.429600001E-03=H(143)
H(59)=-0.226560000E-02=H(142)
H(60)=-0.476490001E-02=H(141)
H(61)=-0.666561000E-02=H(140)
H(62)=-0.667210002E-02=H(139)
H(63)=-0.407420001E-02=H(138)
H(64)= 0.969799999E-03=H(137)
H(65)= 0.731879998E-02=H(136)
H(66)= 0.131523000E-01=H(135)
H(67)= 0.165487001E-01=H(134)
H(68)= 0.161580000E-01=H(133)
H(69)= 0.117259000E-01=H(132)
H(70)= 0.426839999E-02=H(131)
H(71)=-0.419529998E-02=H(130)
H(72)=-0.112830000E-01=H(129)
H(73)=-0.150545000E-01=H(128)
H(74)=-0.146985000E-01=H(127)
H(75)=-0.108397000E-01=H(126)
H(76)=-0.532930002E-02=H(125)
H(77)=-0.566800000E-03=H(124)
H(78)= 0.143520000E-02=H(123)
H(79)=-0.115400000E-03=H(122)
H(80)=-0.434859988E-02=H(121)
H(81)=-0.892230003E-02=H(120)
H(82)=-0.108143000E-01=H(119)
H(83)=-0.748830001E-02=H(118)
H(84)= 0.196160000E-02=H(117)
H(85)= 0.161174000E-01=H(116)
H(86)= 0.313039001E-01=H(115)
H(87)= 0.424900002E-01=H(114)
H(88)= 0.448100002E-01=H(113)
H(89)= 0.352589000E-01=H(112)
H(90)= 0.139929000E-01=H(111)
H(91)=-0.152172000E-01=H(110)
H(92)=-0.456465002E-01=H(109)
H(93)=-0.692685002E-01=H(108)
H(94)=-0.785804012-01=H(107)
H(95)=-0.707509362-01=H(106)
H(96)=-0.450348002-01=H(105)
H(97)=-0.697320002-02=H(104)
H(98)= 0.347401022-01=H(103)
H(99)= 0.700597022-01=H(102)
H(100)= 0.902497022-01=H(101)

B.4 FILTER COEFFICIENTS FOR BAND 4 OF THE 16. KB/S SUB-BAND CODER

FILENAME IS FILT4.NUM

H(1)=-0.12288000E-02=H(200)
H(2)= 0.77679998E-03=H(199)
H(3)=-0.54540002E-03=H(198)
H(4)=-0.14434000E-02=H(197)
H(5)=-0.78250002E-03=H(196)
H(6)= 0.16557000E-02=H(195)
H(7)= 0.39324001E-02=H(194)
H(8)= 0.32953001E-02=H(193)
H(9)=-0.62800001E-03=H(192)
H(10)=-0.46031000E-02=H(191)
H(11)=-0.47221002E-02=H(190)
H(12)=-0.68340002E-03=H(189)
H(13)= 0.36281000E-02=H(188)
H(14)= 0.42984998E-02=H(187)
H(15)= 0.15696000E-02=H(186)
H(16)=-0.92090003E-03=H(185)
H(17)=-0.79580001E-03=H(184)
H(18)= 0.39599998E-03=H(183)
H(19)=-0.20270000E-03=H(182)
H(20)=-0.24871000E-02=H(181)
H(21)=-0.32752000E-02=H(180)
H(22)=-0.63670002E-03=H(179)
H(23)= 0.31417001E-02=H(178)
H(24)= 0.41160001E-02=H(177)
H(25)= 0.16048000E-02=H(176)
H(26)=-0.10787999E-02=H(175)
H(27)=-0.10734000E-02=H(174)
H(28)= 0.28499999E-03=H(173)
H(29)=-0.28440001E-03=H(172)
H(30)=-0.29750001E-02=H(171)
H(31)=-0.40515000E-02=H(170)
H(32)=-0.83520001E-03=H(169)
H(33)= 0.41044001E-02=H(168)
H(34)= 0.55934000E-02=H(167)
H(35)= 0.22684999E-02=H(166)
H(36)=-0.17112000E-02=H(165)
H(37)=-0.21420000E-02=H(164)
H(38)=-0.17990000E-03=H(163)
H(39)=-0.27840000E-03=H(162)
H(40)=-0.33893001E-02=H(161)
H(41)=-0.49712998E-02=H(160)
H(42)=-0.10813000E-02=H(159)
H(43)= 0.54755998E-02=H(158)
H(44)= 0.77582002E-02=H(157)
H(45)= 0.32867000E-02=H(156)
H(46)=-0.26769999E-02=H(155)
H(47)=-0.38792000E-02=H(154)
H(48)=-0.10344000E-02=H(153)
H(49)=-0.20179999E-03=H(152)
H(50)=-0.36545000E-02=H(151)
H(51)=-0.59727002E-02=H(150)
H(52)=-0.13005000E-02=H(149)
H(53)= 0.72780000E-02=H(148)
H(54)= 0.10724400E-01=H(147)
H(55)= 0.47391001E-02=H(146)
H(56)=-0.40942002E-02=H(145)

B.4 (cont'd)

H(57)=-0.656280001E-02=H(144)
H(58)=-0.246579999E-02=H(143)
H(59)=-0.307000001E-04=H(142)
H(60)=-0.368659999E-02=H(141)
H(61)=-0.708319999E-02=H(140)
H(62)=-0.176360000E-02=H(139)
H(63)= 0.974090002E-02=H(138)
H(64)= 0.149625000E-01=H(137)
H(65)= 0.689500001E-02=H(136)
H(66)=-0.626569999E-02=H(135)
H(67)=-0.[illegible]E-01=H(134)
H(68)=-0.[illegible]E-02=H(133)
H(69)= 0.298100000E-03=H(132)
H(70)=-0.332399999E-02=H(131)
H(71)=-0.842779971E-02=H(130)
H(72)=-0.231130002E-02=H(129)
H(73)= 0.135153000E-01=H(128)
H(74)= 0.217576011E-01=H(127)
H(75)= 0.[illegible]E-01=H(126)
H(76)=-0.[illegible]E-01=H(125)
H(77)=-0.185828999E-01=H(124)
H(78)=-0.950040021E-02=H(123)
H(79)= 0.970300000E-03=H(122)
H(80)=-0.[illegible]E-02=H(121)
H(81)=-0.104526001E-01=H(120)
H(82)=-0.320839999E-02=H(119)
H(83)= 0.200073001E-01=H(118)
H(84)= 0.356017021E-01=H(117)
H(85)= 0.181814991E-01=H(116)
H(86)=-0.184841991E-01=H(115)
H(87)=-0.370008011E-01=H(114)
H(88)=-0.211831001E-01=H(113)
H(89)= 0.283750011E-02=H(112)
H(90)= 0.190810001E-02=H(111)
H(91)=-0.156008001E-01=H(110)
H(92)=-0.627310022E-02=H(109)
H(93)= 0.456856001E-01=H(108)
H(94)= 0.888540971E-01=H(107)
H(95)= 0.521575991E-01=H(106)
H(96)=-0.626944011E-01=H(105)
H(97)=-0.156445101E+00=H(104)
H(98)=-0.123516201E+00=H(103)
H(99)= 0.291049001E-01=H(102)
H(100)= 0.170992201E+00=H(101)

B.5 FILTER COEFFICIENTS FOR BAND 5 OF THE 16. KB/S SUB-BAND CODER

FILENAME IS FILTS.NUM

```
H(  1)=-0.606660002E-03=H(200)
H(  2)= 0.94860001E-03=H(199)
H(  3)=-0.273799998E-03=H(198)
H(  4)= 0.170250000E-02=H(197)
H(  5)= 0.187260001E-02=H(196)
H(  6)=-0.287059998E-02=H(195)
H(  7)=-0.255200001E-02=H(194)
H(  8)= 0.316090000E-02=H(193)
H(  9)= 0.346660000E-02=H(192)
H( 10)=-0.380959998E-02=H(191)
H( 11)=-0.383170000E-02=H(190)
H( 12)= 0.385800000E-02=H(189)
H( 13)= 0.349420001E-02=H(188)
H( 14)=-0.309790000E-02=H(187)
H( 15)=-0.237099998E-02=H(186)
H( 16)= 0.161510000E-02=H(185)
H( 17)= 0.672200000E-03=H(184)
H( 18)= 0.253700001E-03=H(183)
H( 19)= 0.116450000E-02=H(182)
H( 20)=-0.190410001E-02=H(181)
H( 21)=-0.258650000E-02=H(180)
H( 22)= 0.302230001E-02=H(179)
H( 23)= 0.310499998E-02=H(178)
H( 24)=-0.298040001E-02=H(177)
H( 25)=-0.248320000E-02=H(176)
H( 26)= 0.179719998E-02=H(175)
H( 27)= 0.854899998E-03=H(174)
H( 28)= 0.192500000E-03=H(173)
H( 29)= 0.127340000E-02=H(172)
H( 30)=-0.232920000E-02=H(171)
H( 31)=-0.315930000E-02=H(170)
H( 32)= 0.382700000E-02=H(169)
H( 33)= 0.406839998E-02=H(168)
H( 34)=-0.404960001E-02=H(167)
H( 35)=-0.354280000E-02=H(166)
H( 36)= 0.276340001E-02=H(165)
H( 37)= 0.160770000E-02=H(164)
H( 38)=-0.271600000E-03=H(163)
H( 39)= 0.117980000E-02=H(162)
H( 40)=-0.263409999E-02=H(161)
H( 41)=-0.386650000E-02=H(160)
H( 42)= 0.490230000E-02=H(159)
H( 43)= 0.542940001E-02=H(158)
H( 44)=-0.559980000E-02=H(157)
H( 45)=-0.513069998E-02=H(156)
H( 46)= 0.426210000E-02=H(155)
H( 47)= 0.284689999E-02=H(154)
H( 48)=-0.113600000E-02=H(153)
H( 49)= 0.813600003E-03=H(152)
H( 50)=-0.282500001E-02=H(151)
H( 51)=-0.464420002E-02=H(150)
H( 52)= 0.623200000E-02=H(149)
H( 53)= 0.721600000E-02=H(148)
H( 54)=-0.771910000E-02=H(147)
H( 55)=-0.738639999E-02=H(146)
H( 56)= 0.645740001E-02=H(145)
```

B.5 (cont'd)

H(57)= 0.47649001E-02=H(144)
H(58)=-0.25804001E-02=H(143)
H(59)= 0.38499998E-04=H(142)
H(60)=-0.28333000E-02=H(141)
H(61)=-0.55078999E-02=H(140)
H(62)= 0.79359002E-02=H(139)
H(63)= 0.96626002E-02=H(138)
H(64)=-0.10744200E-01=H(137)
H(65)=-0.10729500E-01=H(136)
H(66)= 0.98487996E-02=H(135)
H(67)= 0.78319004E-02=H(134)
H(68)=-0.50001998E-02=H(133)
H(69)=-0.14473000E-02=H(132)
H(70)=-0.26313001E-02=H(131)
H(71)=-0.65528998E-02=H(130)
H(72)= 0.10685500E-01=H(129)
H(73)= 0.13422500E-01=H(128)
H(74)=-0.15595300E-01=H(127)
H(75)=-0.16306100E-01=H(126)
H(76)= 0.15889095E-01=H(125)
H(77)= 0.13351100E-01=H(124)
H(78)=-0.96236002E-02=H(123)
H(79)=-0.44968999E-02=H(122)
H(80)=-0.15769000E-02=H(121)
H(81)=-0.81666004E-02=H(120)
H(82)= 0.14800300E-01=H(119)
H(83)= 0.20700793E-01=H(118)
H(84)=-0.25496099E-01=H(117)
H(85)=-0.28278800E-01=H(116)
H(86)= 0.28868500E-01=H(115)
H(87)= 0.26526300E-01=H(114)
H(88)=-0.21312499E-01=H(113)
H(89)=-0.12918300E-01=H(112)
H(90)= 0.16414000E-02=H(111)
H(91)=-0.12244700E-01=H(110)
H(92)= 0.28257995E-01=H(109)
H(93)= 0.45569059E-01=H(108)
H(94)=-0.63603997E-01=H(107)
H(95)=-0.81174999E-01=H(106)
H(96)= 0.97721800E-01=H(105)
H(97)= 0.11202400E+00=H(104)
H(98)=-0.12363850E+00=H(103)
H(99)=-0.13165550E+00=H(102)
H(100)= 0.13583329E+00=H(101)

APPENDIX C

TABULATION OF THE 60-TAP QMF IMPULSE RESPONSE

```
H(  1)=-0.80456801E-10
H(  2)=-0.12466006E-08
H(  3)= 0.33937912E-08
H(  4)= 0.12555994E-07
H(  5)=-0.54866284E-07
H(  6)=-0.79773159E-07
H(  7)= 0.56410033E-06
H(  8)= 0.19332403E-06
H(  9)=-0.40112018E-05
H( 10)= 0.13859838E-05
H( 11)= 0.20943698E-04
H( 12)=-0.18127259E-04
H( 13)=-0.83554602E-04
H( 14)= 0.11246781E-03
H( 15)= 0.26012788E-03
H( 16)=-0.49234246E-03
H( 17)=-0.63178962E-03
H( 18)= 0.16880841E-02
H( 19)= 0.11479852E-02
H( 20)=-0.47692582E-02
H( 21)=-0.12668951E-02
H( 22)= 0.11493831E-01
H( 23)=-0.66263479E-03
H( 24)=-0.24424994E-01
H( 25)= 0.84577305E-02
H( 26)= 0.48056692E-01
H( 27)=-0.31792097E-01
H( 28)=-0.98794423E-01
H( 29)= 0.12183660E+00
H( 30)= 0.46986365E+00
H( 31)= 0.46986365E+00
H( 32)= 0.12183660E+00
H( 33)=-0.98794423E-01
H( 34)=-0.31792097E-01
H( 35)= 0.48056692E-01
H( 36)= 0.84577305E-02
H( 37)=-0.24424994E-01
H( 38)=-0.66263479E-03
H( 39)= 0.11493831E-01
H( 40)=-0.12668951E-02
H( 41)=-0.47692582E-02
H( 42)= 0.11479852E-02
H( 43)= 0.16880841E-02
H( 44)=-0.63178962E-03
H( 45)=-0.49234246E-03
H( 46)= 0.26012788E-03
H( 47)= 0.11246781E-03
H( 48)=-0.83554602E-04
H( 49)=-0.18127259E-04
H( 50)= 0.20943698E-04
H( 51)= 0.13859838E-05
H( 52)=-0.40112018E-05
H( 53)= 0.19332403E-06
H( 54)= 0.56410033E-06
H( 55)=-0.79773159E-07
H( 56)=-0.54866284E-07
H( 57)= 0.12555994E-07
H( 58)= 0.33937912E-08
H( 59)=-0.12466006E-08
H( 60)=-0.80456801E-10
```

659-79E

Appendix D

The Robust Quantizer

The subband coder employs adaptive PCM (APCM) quantizers with the Jayant algorithm which performs well in a back-to-back mode without channel errors. Unfortunately, channel errors significantly impair performance. To see this, consider

$$q_k = M(I_k)\ q_{k-1} = \prod_{i=1}^{k} M(I_i)\ q_o \qquad \text{(D-1)}$$

where I_i is the ith transmitted character and $M(I_i)$ is the multiplier corresponding to I_i.

Assume now that one transmission error is made at time ℓ causing

$$M(I_\ell) \text{ to become } M(I_\ell')$$

and

$$q_\ell \text{ to become } q_\ell' = M(I_\ell')\ q_{\ell-1}$$

afterwards,

the new quantizer levels for time greater than ℓ become

$$q_k' = \prod_{i=1}^{k} M(I_i)\ M(I_\ell')\ q_o \cdot \left[\frac{M(I_\ell)}{M(I_\ell)}\right] \qquad \text{(D-2)}$$

or, simplifying, we obtain

$$q_k' = q_k\ \frac{M(I_\ell')}{M(I_\ell)} \qquad \text{(D-3)}$$

Consequently, the effects of even a single-channel error never die out. In fact, an error causes a level shift in q_k by the factor $M(I_\ell')/M(I_\ell)$. Simulations show that errors cause noticeable fades and increases in output level. Only when the quantizer saturates

at its maximum level or cuts off at its minimum level does the quantizer again track properly.

By incorporating a small decaying factor in the Jayant algorithm, the resulting quantizer can be made more robust to channel errors. This modification calls for the raising of the last quantizer step size, q_{k-1}, to a power β as follows [15]

$$q_k = M(I_k) q_{k-1}^{\beta} = \prod_{i=1}^{k} M(I_i)^{\beta^{k-i}} q_o \qquad \text{(D-4)}$$

where $\pi(\cdot)$ represents the product symbol, I_i is the ith transmitted symbol, and $M(I_i)$ is the multiplier corresponding to I_i. Again, if one channel error occurs at time ℓ

$$M(I_\ell) \rightarrow M(I_\ell')$$

and

$$q_k' = \prod_{\substack{i=1 \\ i=\ell}}^{k} \left[M(I_i) \right]^{\beta^{k-i}} M(I_\ell')^{\beta^{k-\ell}} q_o \cdot \left[\frac{M(I_\ell')}{M(I_\ell)} \right]^{\beta^{k-\ell}} \qquad \text{(D-5)}$$

or simplifying

$$q_k' = q_k \cdot \left[\frac{M(I_\ell')}{M(I_\ell)} \right]^{\beta^{k-\ell}} \qquad \text{(D-6)}$$

Assuming $\beta < 1$, then for times k far removed from ℓ ($k \gg \ell$),

$$q_k' \rightarrow q_k$$

The speed of this convergence depends on how near β is to 1. If β is not too close to unity, convergence is rapid. Consequently, the effect of a single error decays with time.

APPENDIX E

EVALUATING TRANSCENDENTAL FUNCTIONS WITHOUT DIVISION

A. Arcese and A. J. Goldberg

ABSTRACT

We consider least square-error polynomial approximations to trancendental functions over a finite interval. These approximations are useful when computer division times greatly exceed multiplication times. Examples are given for $1/x$, $\sqrt{x}$, e^{-x}, and $\ln x$.

The authors are with GTE Sylvania, Electronics Systems Group, Needham Heights, Massachusetts, 02194.

Given a trancendental function f(x) over some interval (a,b), we seek an m'th degree polynomial approximation to f(x)

$$\hat{f}(x) = \sum_{k=0}^{k=m} a_k x^k \tag{1}$$

that minimizes the sum of squares of the residuals

$$J = \sum_{i=1}^{i=n} \left(f(x_i) - \hat{f}(x_i) \right)^2 \tag{2}$$

over the interval (a,b). The x_i are the n sample data points in the interval (a,b). Minimizing J with respect to each coefficient a_k, we obtain the equation

$$R\alpha = \gamma \tag{3}$$

where the elements of the matrix R are

$$\| R \| = \| \sum_{k=1}^{k=n} x_k^{i+j-2} \| \tag{4}$$

where i, j = 1, 2, . . . , m+1.

The right hand side vector γ has elements

$$\gamma_i = \sum_{k=1}^{k=n} x_k^{i-1} f(x_k) \tag{5}$$

where $i = 1, 2, \ldots, m+1$. α is the solution column vector

$$\alpha^T = \left[a_0, a_1, \ldots, a_m\right] \tag{6}$$

Table I gives the coefficients for a third degree approximation to $1/x$, $\sqrt{x}$, e^{-x}, and $\ln x$ over the interval (0.5, 1.0). For $1/x$ and $\ln x$ it was necessary to subdivide the interval to lower the maximum error. Figures 1 through 5 give the percent error

$$\text{percent error} = \left(\frac{f(x) - \hat{f}(x)}{f(x)}\right) \times 100 \tag{7}$$

for each of the functions.

TABLE I

TABLE OF LEAST SQUARE COEFFICIENTS

$$\hat{f}(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

f(x)	a_0	a_1	a_2	a_3	Range of x
1/x	4.895	-7.918	4.233	0.000	(0.5, 0.75)
	2.626	-1.081	-1.816	1.273	(0.75, 1.0)
$\sqrt{x}$	0.2613	1.116	-0.5193	0.1423	(0.5, 1.0)
e^{-x}	1.060	-1.238	0.7905	-0.2451	(0.5, 1.0)
ln x	-2.222	4.392	-3.167	1.000	(0.5, 0.95)
	-1.007	0.4711	1.071	-0.5358	(0.95, 1.0)

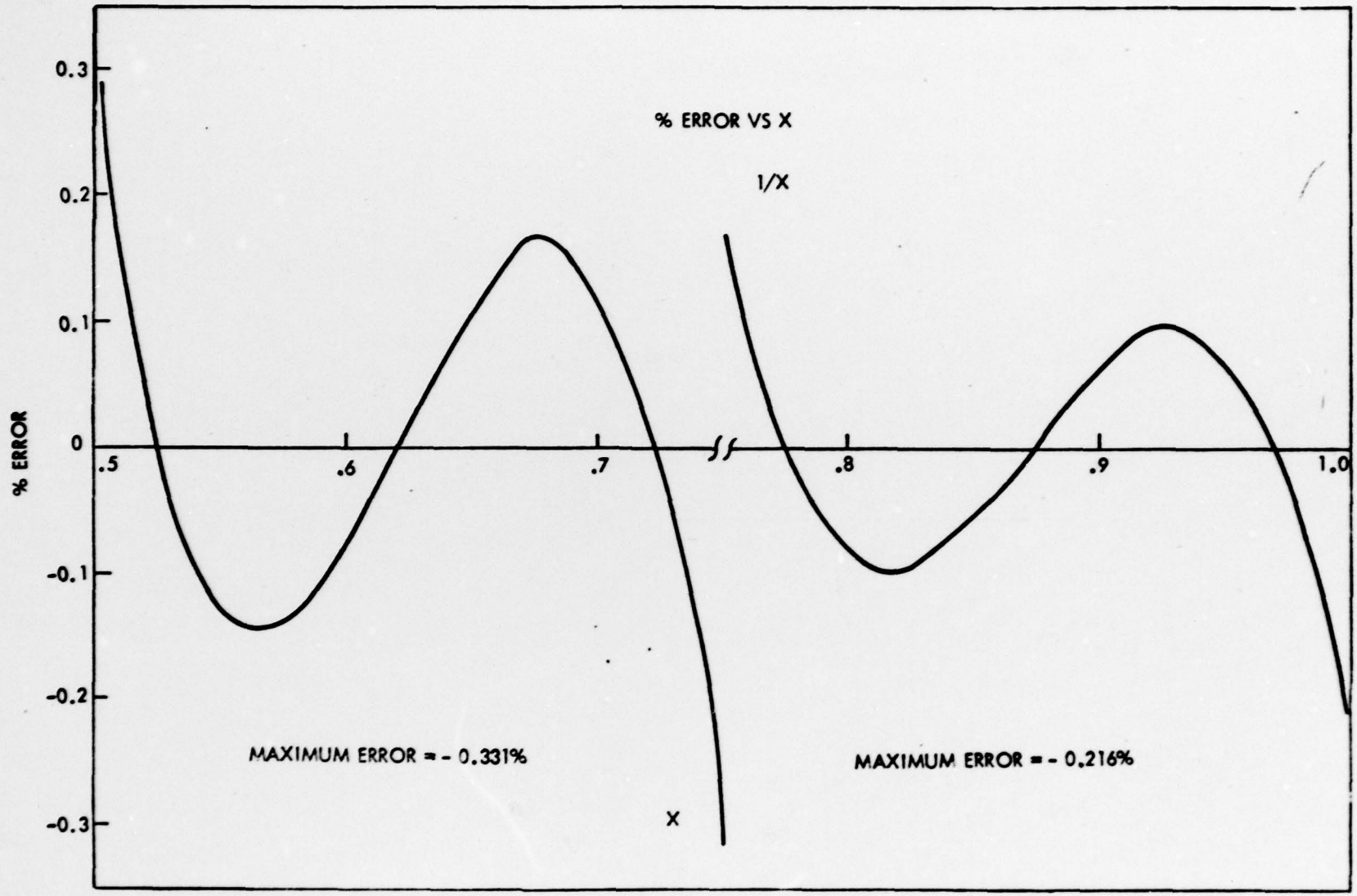

% ERROR VS X
1/X
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
% ERROR
.5
.6
.7
.8
.9
1.0
MAXIMUM ERROR = - 0.331%
MAXIMUM ERROR = - 0.216%
X

FIGURE 1

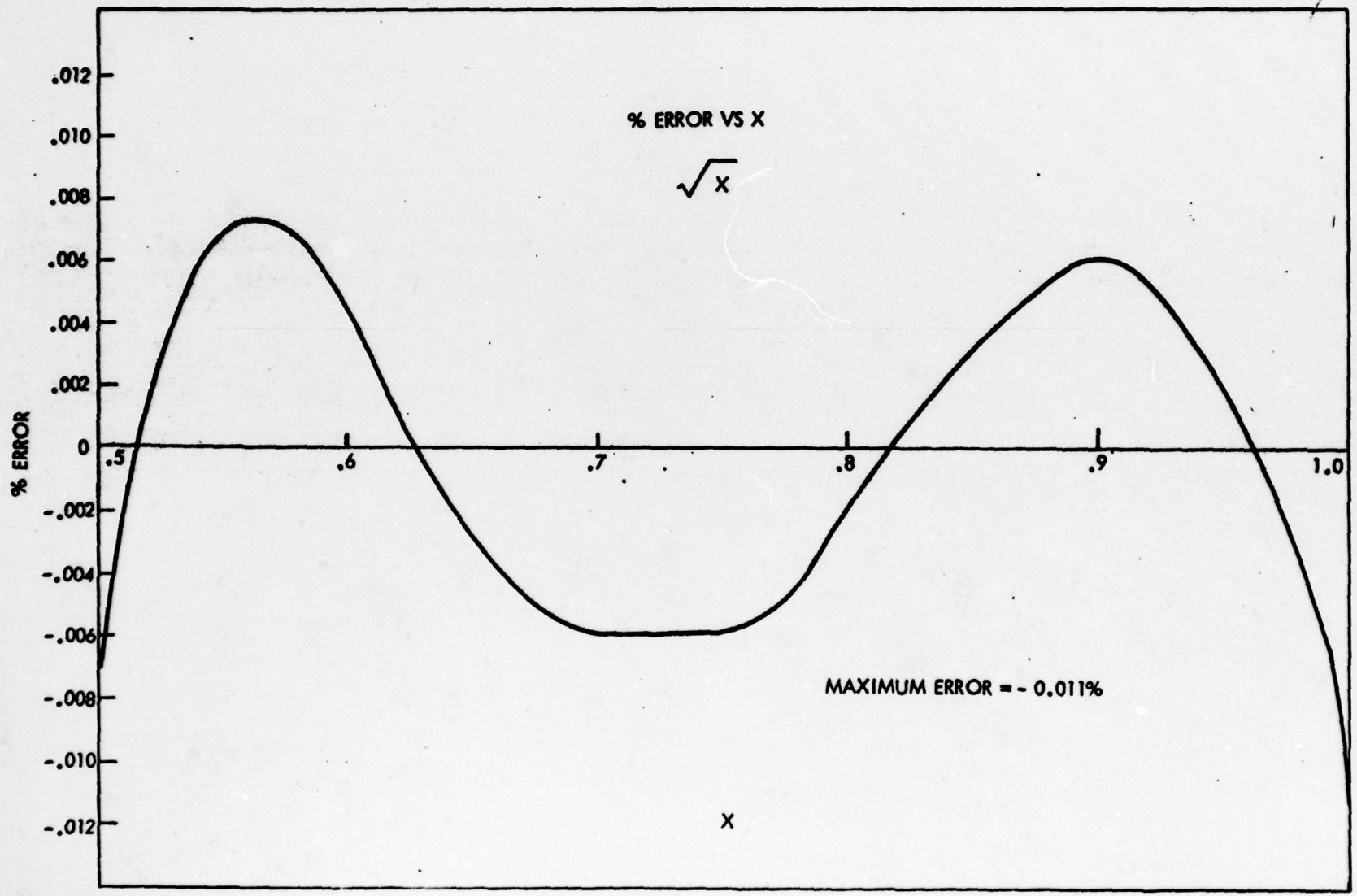

% ERROR VS X
$\sqrt{X}$
.012
.010
.008
.006
.004
.002
0
-.002
-.004
-.006
-.008
-.010
-.012
% ERROR
.5
.6
.7
.8
.9
1.0
X
MAXIMUM ERROR = - 0.011%

FIGURE 2

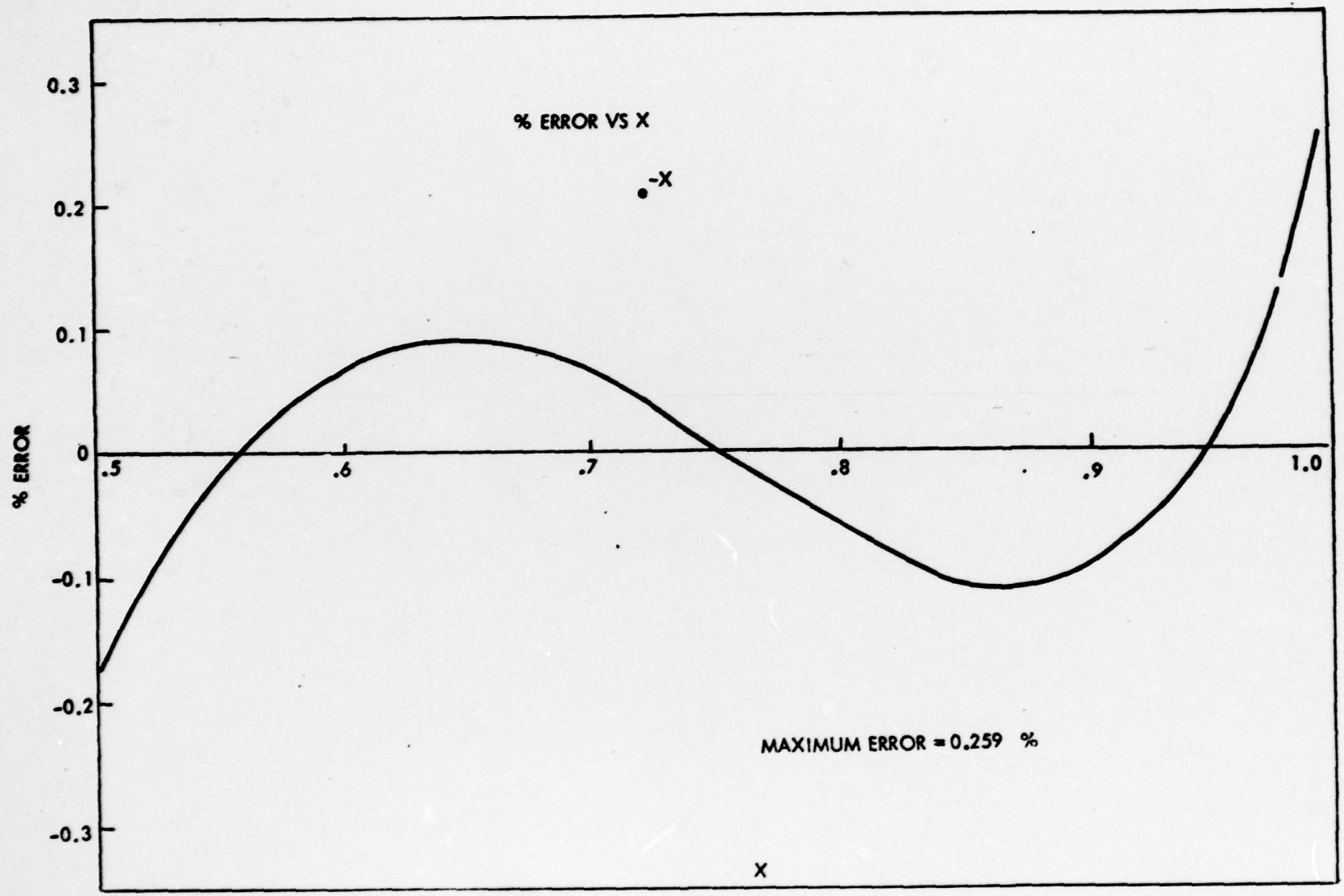

% ERROR VS X
-X
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
% ERROR
.5
.6
.7
.8
.9
1.0
X
MAXIMUM ERROR = 0.259 %

FIGURE 3

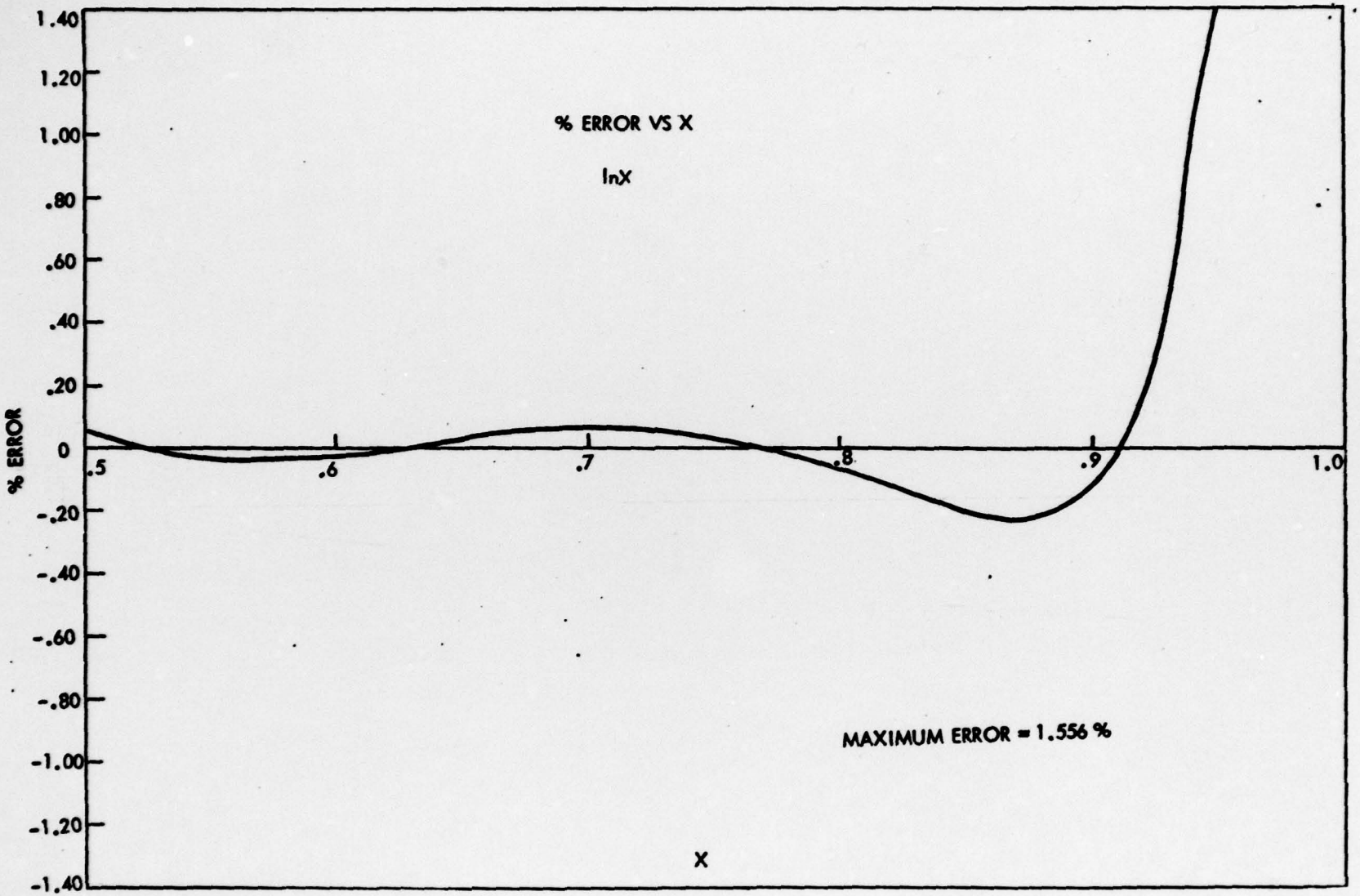

% ERROR VS X
lnX
% ERROR
1.40
1.20
1.00
.80
.60
.40
.20
0
-.20
-.40
-.60
-.80
-1.00
-1.20
-1.40
.5
.6
.7
.8
.9
1.0
X
MAXIMUM ERROR = 1.556 %

FIGURE 4

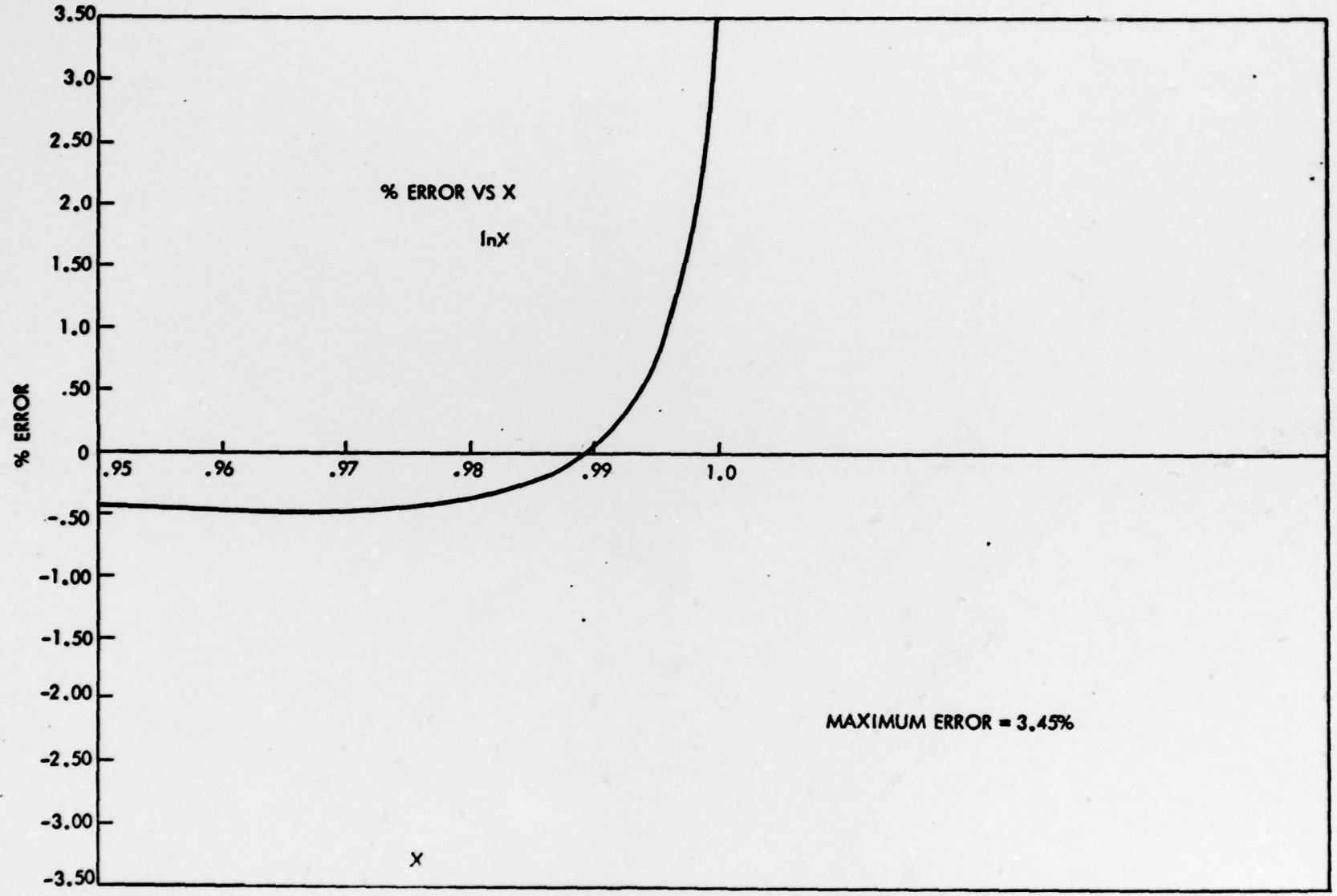

% ERROR VS X
lnX
% ERROR
3.50
3.0
2.50
2.0
1.50
1.0
.50
0
-.50
-1.00
-1.50
-2.00
-2.50
-3.00
-3.50
.95
.96
.97
.98
.99
1.0
X
MAXIMUM ERROR = 3.45%

FIGURE 5